SRL 2147 TR 71-3
Technical Report

A CHARACTERIZATION OF THE VISIBILITY PROCESS AND ITS EFFECT ON SEARCH POLICIES

Michael Leo Moore

SYSTEMS RESEARCH LABORATORY
Department of Industrial Engineering
The University of Michigan

December 1971

Contract No. N00014-67-A-0181-0012

Office of Naval Research
Department of the Navy

FOREWORD

Under Contract No. N00014-67-0181-0012 with the Office of Naval Research, the Systems Research Laboratory (SRL) has been conducting a research program to develop analytic models of defense processes, principally the combat process. A detailed description of all the research performed on this program through June 1970 was reported in SRL 2147 TR 70-2 (U) "Development of Models for Defense Systems Planning" dated September 1970. Additional work related to the combat allocation process was reported in SRL 2147 TR 71-1 (U) "Development of Optimal Strategies in Heterogeneous Lanchester-Type Processes" dated June 1971.

The work in descriptive modeling of combat processes and the development of optimal weapon allocation strategies assumed perfect intelligence gathering capabilities of the forces. For this reason some of the research effort has been directed to the study of intelligence and reconnaissance processes. A literature review of this area (reported in SRL 2147 TR 70-1 "A Review of Search and Reconnaissance Theory Literature", dated January 1970) indicated the need to consider more realistically both environmental effects and search objectives (interaction with the combat process) in developing descriptive structures of the search process and analysis of optimal search

strategies. Some initial ideas in these directions were presented in SRL 2147 TR 70-2 (U). Initial research on incorporating the effects of search objectives (called the response process) has been performed and is described in the report SRL 2147 TR 71-2 (U) "Effects of the Response Process in Search Models with False Detections."

The work described in ~~this~~ report considers an important environmental effect -- the visibility process. The research explores the development of mathematical structures which link the detection and visibility processes, examines the effect that the visibility process has on classical search strategies, and provides guidelines regarding search situations which require explicit consideration of the visibility process in the development of optimal search policies.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF TABLES

Chapter 1

INTRODUCTION AND LITERATURE REVIEW

The topic of this report arose out of research performed by the Systems Research Laboratory (SRL) under contract number N00014-67-A-0181-0012 with the Office of Naval Research. The overall research program is concerned with the development of more generalized mathematical structures of military processes, although as will soon become evident, the results are applicable to nonmilitary processes as well. Emphasis has been directed to descriptive modeling of combat processes and has assumed perfect intelligence-gathering capabilities of the units involved. It was thought that many of the existing search and reconnaissance theories would be useful for predicting the amount of intelligence-gathering capability processed by a tactical unit. However, a thorough literature review indicated that most search and reconnaissance modeling efforts have been devoted to the development of strategies for the optimal allocation of search effort and little to the development of realistic descriptive models of intelligence-gathering process. The literature review and preliminary modeling efforts (Moore, 1970) indicated the need to consider more realistically both environmental effects and search objectives in developing

descriptive structures of the search process and analysis of optimal search strategies. This report adresses one dimension of the environmental effects--the "visibility process." The main thrust of the work is th velopment of descriptive search models which include the visibility process, and the analysis of these models to gain physical insight into the intelligence-gathering process, and to determine strategies for optimal allocation of search effort.

In this chapter, some fundamental terms are defined and a taxonomy of possible classifications of search problems is introduced in order to establish a common vocabulary. A classification of search problems is then introduced to familiarize the reader with the general structure of search problems. Next the results of an extensive literature review are summarized to provide the reader with background in the area as well as a ready source of those results which are referred to in the body of the thesis. Finally, the specific area of research is outlined and the broad applicability of the visibility models discussed.

1.1 Characterization of the Elements of Search Problems

1.1.1 Definitions

This section contains some basic definitions and notations used throughout the paper. Any additional notation and exceptions to those specified herein will be specifically noted in the text.

Detection - The act of gathering information pertaining to the object being sought, the sifting out of what is important information and the relaying of that information in some efficient form to the decision maker.

Discrete Detection Model - Let $(1 - q_i)$ be the instantaneous probability of detection of the i^{th} scan of an area. Given n such scans, the probability of detection is

$$P(D) = 1 - \prod_{i=1}^{n} q_i \ .$$

The q_i are referred to as "overlook" probabilities.

Continuous Detection Model - The probability of detecting the target in the interval $(t, t + dt)$, given no detection up to t, is given by $\gamma(t)dt$. Given continuous observance over an interval $(0,\tau)$, the probability of detection is

$$P(\tau) = 1 - e^{-\int_0^{\tau} \gamma(t)dt} \ .$$

Search Strategy - The decision made on the basis of information obtained from the detection process. A "search strategy" will be that set of rules which associates a decision with every conceivable result of the detection process, e.g., the next region to be searched and how much effort to expend there.

Target - The object of the search: a military target, a mineral deposit, or any other object about which information is desired.

Search Space - The region containing the target or targets. It can be *discrete* consisting of boxes or subregions, or it may be *continuous*. It need not be considered in a strictly geometrical sense, e.g., the search space may consist of the possible frequencies of an unknown signal.

Cumulative Detector - A detection device having either of the following characteristics:

(a) No loss of information - Given that x units of effort have been continuously applied, all the information thus gained is retained, when at some later time, additional effort is placed in the same region.

(b) Partial loss of information - Given that x units of effort have been continuously applied, only certain portions of the information thus gained are retained when at some later time, additional effort is placed in the region. The portion retained could be a function of:

(1) the length of time since the first trial

(2) for moving targets, the motion structure.

Non-Cumulative Detector - A detection device having the property that all the information gained from a previous search is lost, when at some later time additional effort is placed in the same region.

Visibility - That condition under which the sensor (detector) signals can reach the target and be received.

Search Objectives - The two common search objectives are:

(a) Given a constraint on the available search effort, maximize the probability of detecting the target.

(b) Given unlimited search effort, minimize the expected effort required until detection occurs.

The first is an effectiveness measure, while the second is a cost measure.

1.1.2 Classification of Detectors and Targets

Models of search and reconnaissance processes treat detectors and targets with varied combinations of properties or assumptions regarding their behavior. This section presents a classification of analytic assumptions that can be used to describe the behavior of detectors and targets.

Detectors

1. Single Detector with a Single Scan.[1]
   A) Discrete detection
   B) Continuous detection

---

[1] Aerial photographic reconnaissance provides an example of this situation.

1) Non-cumulative probability of detection[1] (Complete loss of information)

2) Cumulative probability of detection

a) partial loss of information

b) no loss of information

2. Single Detector with Multiple Scan Capability

A) Discrete detection

B) Continuous detection

1) Non-cumulative probability of detection[1]

2) Cumulative probability of detection

a) partial loss of information

b) no loss of information

3. Multiple Detectors with Single Scan Capability

A) Discrete detection

1) Detectors act statistically independently

2) Detectors act dependently

B) Continuous detection

1) Independent action

a) non-cumulative probability of detection[1]

b) cumulative probability of detection

(1) partial loss of information

(2) no loss of information

2) Dependent action

---

[1] These conditions can also hold for a discrete detector.

a) non-cumulative probability of detection

b) cumulative probability of detection

(1) partial loss of information

(2) no loss of information

4. Multiple Detectors with Multiple Scan Capability

A) Discrete detection

1) Independent action

2) Dependent action

B) Continuous detection

1) Independent action

a) non-cumulative probability of detection[1]

b) cumulative probability of detection

(1) partial loss of information

(2) no loss of information

2) Dependent action

a) non-cumulative probability of detection

b) cumulative probability of detection

(1) partial loss of information

(2) no loss of information

Targets

1. Non-Conscious Evasion

A) Single target

1) Stationary

a) continuously visible

b) not continuously visible

[1]These conditions can also hold for a discrete detector.

2) Moving

a) continuously visible

b) not continuously visible

B) Multiple Targets

1) Stationary

a) number

b) location

(1) independent actions

(a) continuously visible

(b) not continuously visible

(2) dependent actions

(a) continuously visible

(b) not continuously visible

2) Moving

a) number

b) initial distribution; description of the motion

(1) independent actions

(a) continuously visible

(b) not continuously visible

(2) dependent actions

(a) continuously visible

(b) not continuously visible

2. Conscious Evasion

(Same as 1 above)

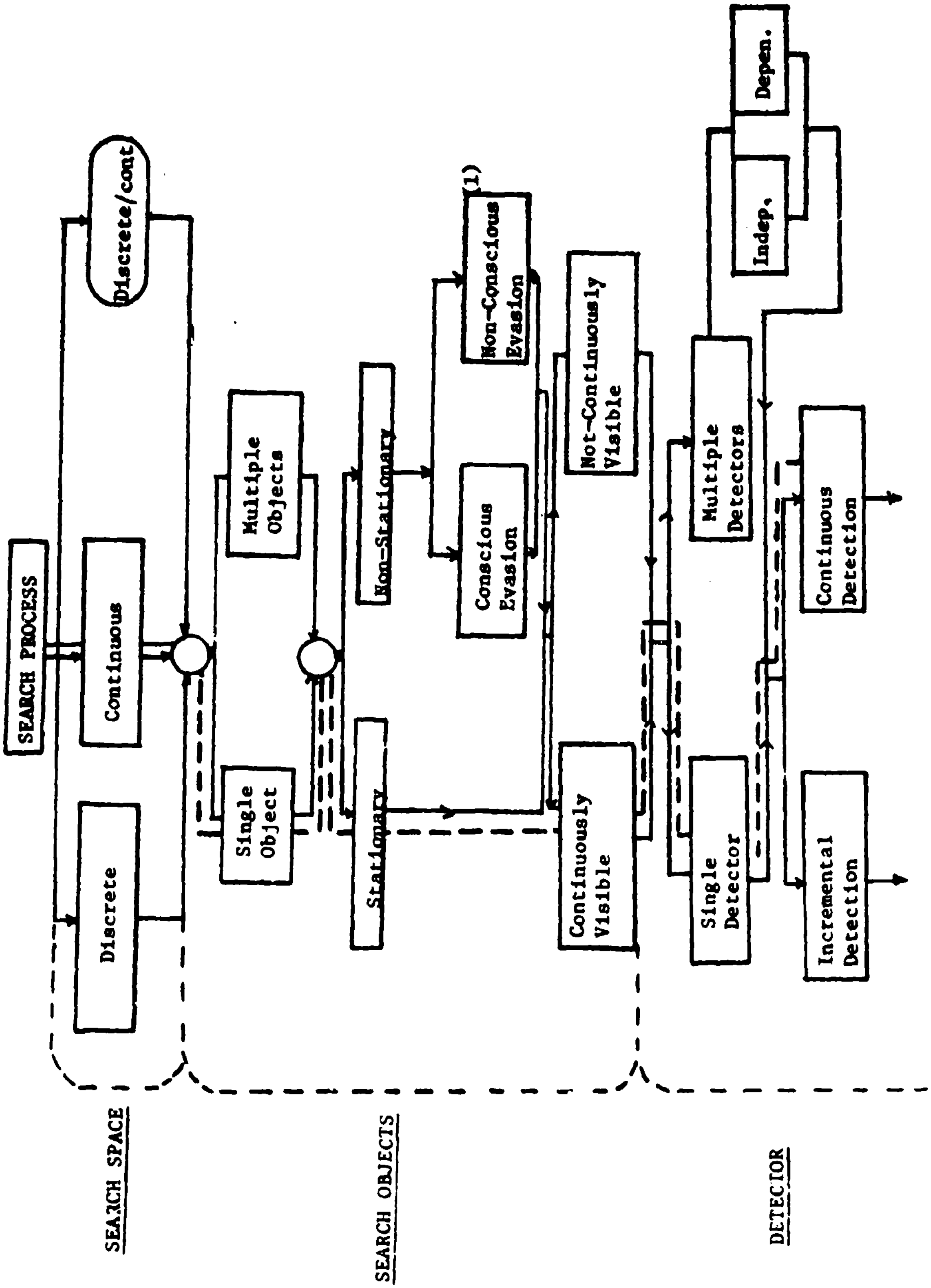
1.1.3 Classification of Search Problems

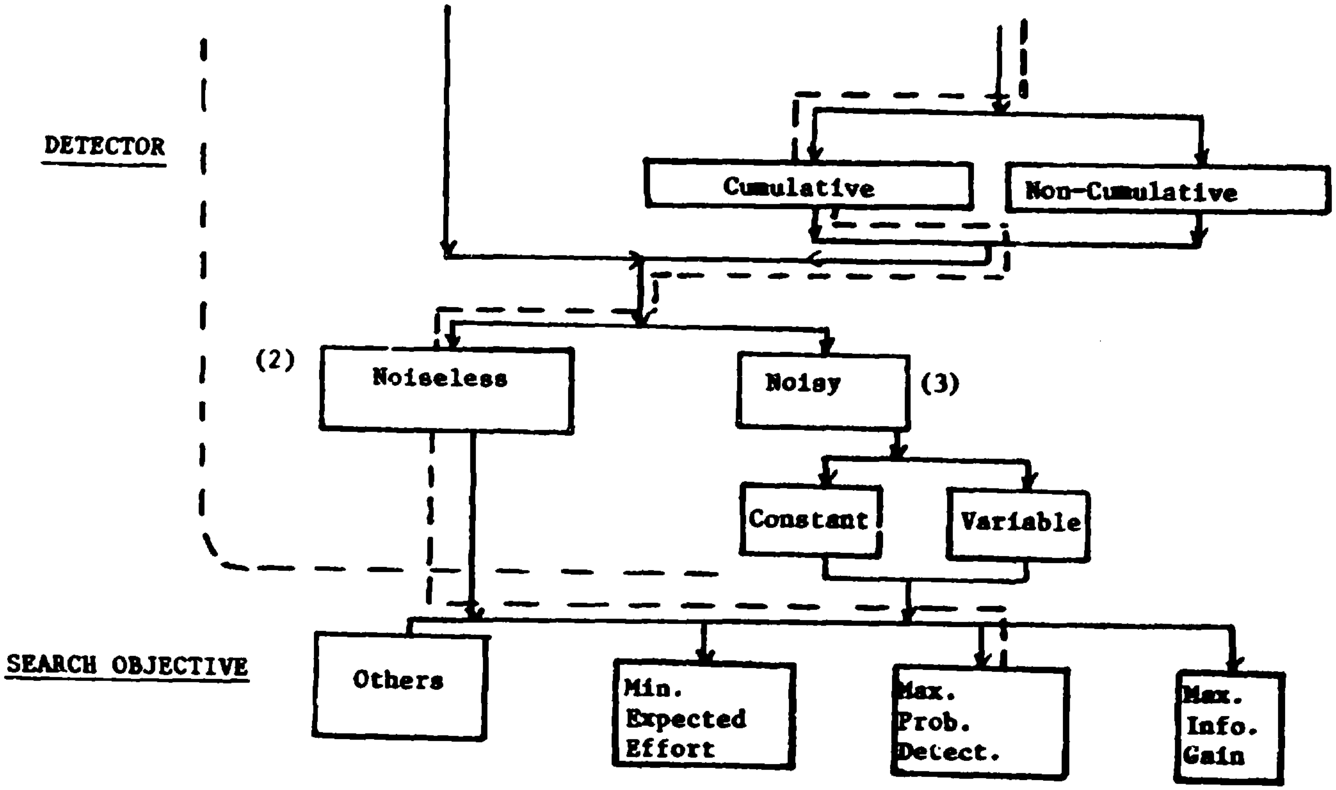
Many variations of search problems appear in the open literature. The following decision diagram[1] presents, in flow chart format, the various attributes of search problems that may be considered. It is introduced at this point in the paper to familiarize the reader with the general structure of search problems and to provide a vehicle for the location of the proposed research topics within the structure. Given any of the problems discussed in the literature, one can characterize it by a path through the decision diagram. For example, the classic search problem of Koopman (1946) which involves a single searcher with a continuous detection device looking for a stationary target, continuously visible, located on the real line is characterized by the dotted path.

1.2 Literature Review

The author has compiled a reasonably complete review of the open literature in search theory, (Moore, 1970), in an effort to provide a base for the research performed in this study and to indicate fruitful areas of research for other investigators. This section summarizes the results of the review. The following classification scheme will be

[1]The chart given is a modification of one originally given by H. Beiman, "An Investigation of Sequential Search Algorithms," Oper. Res. Inc., Silver Spring, Maryland, AD 657050, January, 1967.


SEARCH PROCESS
SEARCH SPACE
Discrete
Continuous
Discrete/cont
SEARCH OBJECTS
Single Object
Multiple Objects
Stationary
Non-Stationary
Conscious Evasion
Non-Conscious Evasion
(1)
Continuously Visible
Not-Continuously Visible
DETECTOR
Single Detector
Multiple Detectors
Indep.
Depen.
Incremental Detection
Continuous Detection



---

(1) Includes the case of perfect information on the part of the searcher, i.e., searcher is aware of all the strategies available to the target.

(2) Perfect detector, (i.e., no false alarms).

(3) Includes false alarms with the possibility of a variable false alarm rate.

utilized to outline the historical development of what we shall term "classical search theory:"

(A) Non-Conscious Evasion

These papers deal with the problem of determining the optimal allocation of effort to find a target when the probability distribution of the target location is known to the searcher

Two sub-headings are considered under this category.

(1) Stationary targets, i.e., the target is assumed stationary although some authors consider targets that suddenly appear and remain visible.

(2) Moving targets, i.e., the target is moving without conscious evasion and the searcher knows, or is willing to assume, the motion or distribution of motion of the target.

(B) Conscious Evasion

These papers, which usually include game theoretic concepts, consider the search problem with a conscious evader. Included in this category are the search/evasion problems in which the searcher and evader can alter their motions differentially by choices of continuously varying parameters.

(C) Search and the Visibility Process

This section includes papers containing important results, utilized in our work, in the development of search and detection theory and methodology or the application of search concepts to the operations of reconnaissance and surveillance.

1.8.1 Non-Conscious Evasion

Koopman (1946) laid the foundation for the entire field of optimal search and detection. In a series of fundamental papers he described certain basic detection processes and the procedure for the optimum allocation of effort in the search for a stationary target. We shall define this problem and its solution in some detail in order to better understand the later developments in the field, and to make some comparisons with our results.

Let a stationary target be located in a known region A with known complete probability density function $p(x,y)$ continuous over A. It is assumed that the searcher has certain constraints on the amount of effort, Φ, that can be directed towards the search. Let $\phi(x,y)$ be a search density function defined on the region A, with the properties that

$$\iint_A \phi(x,y)dxdy = \Phi ,$$

$$\phi(x,y) \geq 0 \text{ on } A .$$

If one assumes that the searcher is employing a continuous detector, then the probability of detecting the target with effort ϕ, $P(\phi)$, is given by

$$P(\phi) = \iint_A p(x,y)\left(1 - e^{-\phi(x,y)}\right)dxdy ,$$

where $1 - e^{-\phi(x,y)}$ represents the conditional probability of detecting a target at (x,y) with effort $\phi(x,y)$, conditioned on the target being present, and is derived as a consequence of Koopman's Law of Random Search (Koopman, 1946).

The fundamental problem is to determine from the class of functions satisfying the constraints that search density function which maximizes the objective function. Koopman obtained the result:[1]

$$\phi(x,y) = \begin{cases} \log p(x,y) - \frac{1}{\hat{A}} \iint_{\hat{A}} \log p(x,y)dxdy + \frac{\Phi}{\hat{A}}, & (x,y)\varepsilon\hat{A} \\ 0 , & (x,y)\varepsilon A - \hat{A} \end{cases}$$

where

$$\hat{A} = \left\{ (x,y):p(x,y) \geq b \;\&\; \log b - \frac{1}{\hat{A}} \iint_{\hat{A}} \log p(x,y)dxdy + \frac{\Phi}{\hat{A}} = 0 \right\} .$$

By considering A to be composed of subregions $A_1, A_2, \ldots, A_N$, one can obtain the solution to the N-region discrete search space problem. Some generalizations suggested by Koopman

---

[1]The solution gives one the optimal allocation of the effort. One is not concerned here with the sequence of allocations.

include the case of visibility varying from position to position, i.e., let v(x) denote the probability that a target in (x,x + dx) is visible, weighting the probability of detection by a function dependent upon where the target is detected, e.g., the detection of saboteurs, and weighting the search density function by a cost function dependent upon the region being searched.

Charnes and Cooper (1958) developed an algorithm for the solution of a discrete search space/continuous detector version of Koopman's problem, i.e., the search region is divided into n subregions with an a priori probability vector on the target position. The algorithm is obtained from the application of the Kuhn-Tucker conditions for optimality to the resulting convex programming problem.

Blackman (1959) considered the following discrete search space variation on the Koopman problem. The target is not present at the beginning of the search, but the searcher has a prior distribution of arrival times, and the objective is not the maximization of the probability of detection, but minimization of the expected time to detect the target after arrival. If the time of appearance of the target is uniformly distributed over a long interval of length T, the author determines the order in which the various possible locations should be scanned to minimize the expected time

between the appearance of the target and its detection[1]. Blackman and Proschan (1959) studied similar problems of objects arriving in accordance with a Poisson process. Having arrived, the objects appear (and remain until detected) in region i with probability p_i. A single scan of the i^{th} region costs c_i, takes time t_i, and, if the object is present at the start of the scan, will detect it with probability $(1-q_i)$. The resultant gain $g_i(t)$ is a non-increasing function of t, the time between the arrival and the beginning of the detecting look. Considering only cyclic search schedules, i.e., those which repeat after D units of time, where D is arbitrary, the authors derive the optimum search schedule.

Gilbert (1959) considers a two-region search problem, with continuous detection and including nonzero switching times, with the objective of minimizing the expected time until detection. His results are summarized in two theorems: The first gives necessary conditions on the optimum solution (sequence and allocations); the second, upper and lower bounds on the value of the objective function. In the special case in which the detection probabilities satisfy the Law of Random Search and are identical, the first theorem will in fact yield the optimal solution.

---

[1]Here the objective is to determine the optimal *sequence*, i.e., a sequential rather than a parallel search problem.

MacQueen and Miller (1960) deal with the problem of whether or not a search activity should be started, and, if started, whether or not it should be continued. Their model gives rise to a general functional equation for which existence and uniqueness conditions are given.

Pollock (1960) introduces a discrete detection model for the two-region search problem and determines the optimal sequential strategies for this model. As the search progresses, the a posteriori probabilities of target position are obtained using Bayes' Theorem. Switching rules are derived for the search sequence which minimizes the expected length of search. The author makes some comparisons of the optimal values of the expected length of the search for discrete and continuous detection models, deriving the conditions for similarity. He also makes the important observation that, *for the models under consideration*, the criteria of (a) maximizing the probab detection by the end of a fixed time and (b) mini he expected length of time until detection; both lead e same results in terms of the allocation of effort.

The detection processes in Koopman's formulations were restricted to those which satisfied the Law of Random Search; de Guenin (1961) generalized these processes. He made the following assumptions regarding $P(\phi)$, the probability of detecting the target with an effort $\phi(x)$ when the target is at x:

(1) $P(0) = 0$

(2) $\frac{dP(\phi)}{d\phi} = P'(\phi) \geq 0$

(3) $P'(\phi)$ is a decreasing, continuous function of ϕ

(4) $P'(0) > 0$, $P'(\infty) = 0$.

From the above properties, $P'(\phi)$ has an inverse function $\phi = f(P')$.

The basic problem becomes

$$\max \int_{-\infty}^{\infty} p(x) \; P[\phi(x)]dx ,$$

$$\text{S.T.} \int_{-\infty}^{\infty} \phi(x)dx = \Phi$$

$$\phi(x) \geq 0$$

where $p(x)$ is the probability density function for the target location. de Guenin's major result is: "whenever the distribution of effort is optimum, the marginal effort required to increase the detection probability at any point is proportional to the probability density, $p(x)$, of the location of the object."

Zahl (1963) considered the following general class of problems:

$$\max \int f[x,y(x)]dx$$

$$\text{S.T.} \int g[x,y(x)]dx = \text{constant}$$

$$a(x) \leq y(x) \leq b(x) .$$

He gives necessary and sufficient conditions for a maximizing function y(x) under fairly weak conditions. One may readily interpret the above problem in the search theory context, i.e., with the notation used for de Guenin, take

$$\phi(x) = y(x) ,$$

$$f[x,y(x)] = g(x)P[\phi(x)] ,$$

$$g[x,y(x)] = \phi(x) .$$

The author requires only that the conditional detection function $P(\phi)$ be nondecreasing in $\phi(x)$.

Dobbie (1963) develops sufficient conditions for Koopman's additive property to hold. This property states that the distribution which maximizes the detection probability with a given amount of effort, ϕ, has the interesting property that it is the sum of the optimal distribution of effort ϕ_1 and the conditionally optimal distribution of effort ϕ_2, $(\phi_1 + \phi_2 = \phi)$ given that the target has not been found with the previous distribution of effort ϕ_1. The sufficient condition requires that the conditional detection function be an increasing concave function of the search density

function. The author also points out that one can express the expected effort required to detect the target as

$$E = \int_0^\infty Q(\phi)d\phi = \int_0^\infty (1 - P(\phi))d\phi$$

where $P(\phi)$ is the probability of detecting the target with effort distributed according to a particular effort density function. From this equation, one can see that the expected effort, e.g., the expected time to detect the target, is minimized by always distributing the effort to maximize the probability of detection with the effort expended thus far. This suggests, under the conditions of the concave objective function, that the strategies for both problems are identical.

Pollock (1964) develops search strategies to minimize the expected cost of search. The search process is represented in terms of a stochastic dynamic program. The optimal search strategies as well as the associated minimum costs are given. It is shown that the optimal solution is the Wald sequential probability ratio test.

Matula (1964) derives conditions for the existence of an "ultimately periodic" search program in the following context. An object is located in one of a finite number of possible locations with a priori probability p_i . Associated with each location i is a cost for searching that location, c_i, and an overlook probability q_i. The problem is to determine a sequence of locations to be searched such that the

expected cost of finding the object is minimized. A program is called ultimately periodic if after a transient period of length T, the sequence of locations to be searched is repeated with the length of the period denoted by θ. The author obtains the following theorem.

Periodic Search Theorem

A necessary and sufficient condition for the existence of an ultimately periodic optimal program is that the ratios $[\log q_i/\log q_j]$ all be rational.

It is interesting to note that the limiting frequency of search of a location for any optimal program depends only on the overlook probabilities, not on the initial probability distribution on target location or even the relative costs.

Kadane (1968) studies the problem of choosing a strategy to maximize the probability of finding a stationary object in a discrete search space when a budget ceiling is imposed. He also assumes that the overlook probability, a discrete detector is assumed, may depend upon the region and the *number of previous looks*. The major result in this paper is the following extension of the Neyman-Pearson Lemma,

Theorem

Let $\{P_i\}$ and $\{C_i\}$ be arbitrary nonnegative sequences such that $\Sigma P_i < \infty$. Let X be the class of sequences X_i, such that $0 \leq X_i \leq 1$, $\forall_i$, then the maximum of

$$\sum X_i P_i$$

$$\text{S.T.} \quad \sum X_i C_i \leq C$$

$$X_i \in X$$

is attained. It occurs when and only when

$$X_i = \begin{cases} 0, & P_i < rC_i \\ 1, & P_i > rC_i \end{cases}$$

for some r, $0 < r < \infty$, and $\Sigma X_i C_i = C$.

The author describes an integer programming algorithm (branch and bound variety) adapted to the above problem. The ratio P_i/C_i is the appropriate decision variable in the sense that searches with large P_i/C_i should be included and those with small P_i/C_i excluded. P_i/C_i plays the role of a cost-effectiveness criterion. Zahl (1963) gave the continuous analog of these results. Black (1965) presents a graphical argument for the optimal sequential search procedure for the minimum cost problem. He shows that the policy with the minimum expected cost is generated by the rule: "Always look in the region for which the posterior probability (given the failure of earlier looks) of finding the object divided by the cost is maximum." Chew (1967) gives the following optimal strategy: to maximize the probability of finding the object in a fixed number, N, of searches, choose those

N searches for which the posterior probability (given the failure of earlier glimpses) of finding the object is largest. It is assumed that the overlook probability does not depend upon the number of previous searches.

Koopman (1946) was the first to examine situations in which a search is to be conducted for a moving target. Among these are the barrier patrol search procedures for a target moving through a straight channel with the vector velocities at all points parallel and equal. Another situation studied is that in which an initial distribution of target location is given at the time of fix, but with the target moving in a random direction with an estimated constant speed. The objective is the construction of a search after a large amount of time has elapsed from the time the target is fixed to the initiation of the search. It is shown that in order to obtain the maximum probability of detection per unit time, the ideal track for a "cookie cutter" detector is an equiangular or logarithmic spiral which is approximated by the "retiring square search procedure."

Klein (1968) considered the following moving-target problem. An object moves about within a finite number of regions, one per time unit, according to the transition matrix

$$H = \{h_{ij}\}$$

where i denotes the searcher's current location and j the target's next, (obviously the target discovers the searcher's location at the end of each period). A single searcher, using a continuous detection system whose effectiveness is a function of the amount of effort used and the region searched, checks one region at a time until the object is found, his budget exhausted, or he decides it is "uneconomical" to continue. The problem is to find an optimal sequential search policy, i.e., one which tells the searcher, at each point in time, whether to search, where to search, and how much effort to use, for the following problems:

(a) Minimize the expected cost subject to achieving a specified level of the probability of detection.

(b) Minimize the expected time until detection subject to achieving a specified level of the probability of detection, and an upper bound on the budget.

(c) Maximize the probability of detection subject to upper bounds on the expected duration of the search and the expected cost.

The author doesn't solve these problems, but suggests that certain of these formulations can be transformed into linear programming problems.

Pollock (1970) considers a target moving in a Markovian fashion between two regions. Dynamic programs for the

standard problems of minimizing the expected time until detection and maximizing the probability of detection under a constraint on search effort are solved. For certain special forms of the transition matrix, decision rules are derived for the minimum expected time problem as well as upper and lower bounds for the minimum expected time.

Wagner (1969) develops the following theorem which is applicable to either continuous or discrete search space problems.

Theorem

Let e and c be real-valued functions (of two variables) defined on

$$\{(x,j) \mid a < x < b,\ \ell(x) \leq j \leq u(x),\ j \text{ an integer}\}\ .$$

Let ϕ be the set of all integer-valued functions f on (a,b) such that $\ell(x) \leq f(x) \leq u(x)$ for $a < x < b$ for which

$$-\infty < E(f) = \int_a^b e(x,f(x))dx < \infty$$

$$-\infty < C(f) = \int_a^b c(x,f(x))dx < \infty\ .$$

Suppose that $g \in \phi$ has the following property: there exists a $\lambda > 0$ such that for all $x \in (a,b)$ and integers j

$$e(x,j) - e(x,j-1) \leq \lambda[c(x,j) - c(x,j-1)] \text{ whenever } g(x) < j \leq u(x)$$

$$e(x,j) - e(x,j-1) \geq \lambda[c(x,j) - c(x,j-1)] \text{ whenever}$$
$$\ell(x) \leq j - 1 < g(x)$$

then

$$E(g) = \max \{E(f) \mid f \in \phi \text{ and } C(f) \leq C(g)$$
$$C(g) = \min \{C(f) \mid f \in \phi \text{ and } E(f) \geq E(g)$$

Onaga (1971) studied the problem of the minimization of the expected time until detection for general detection functions. He also includes penalty or switch times which occur whenever the searcher changes regions. The major results of the paper are summarized in two theorems. The first is a necessary condition for optimality and is applicable to either a continuous or discrete detection function, $P(\phi)$. It consists of two characterizations: the first regulates the optimal lengths of visit times and the second determines the optimal search order. The second result is a necessary and sufficient condition stated in constructive form which is applicable to unimodal probability density functions $\rho(\phi)$. Of special importance to the results of our work is the following theorem.

Theorem 7

If the density function, $\rho(\phi)$, is unimodal, one can use the minimal concave majorant of $\rho(\phi)$ for obtaining the optimal policies.

1.2.2 Conscious Evasion

Norris (1962) appears to have been among the first to consider the two-sided search problem. The search is conducted against a conscious evader who is able to observe the searcher's actions and capitalize on any errors he makes. The evasion device of moving between looks is treated. The game is zero-sum and incorporates a fairly general reward structure which can include discounting. The reward coefficients associated with this structure, as well as the detection probabilities, are known to both players. Three levels of moving costs for the evader are considered for a two-region search problem. In the case of an infinite moving cost, the author derives a condition which is a special case of the Periodic Search Theorem of Matula (1964). The searcher's "good strategy" in the case of a finite moving cost is generated by a finite Markov process. Finally, when no moving cost is incurred by the evader, the searcher cannot gain any information concerning the evader's position from his past sequence of unsuccessful looks. Therefore, each look should be made according to the same probability distribution. In the N-region formulation of the finite moving cost game, a "good search" strategy cannot be generated by a finite Markov process.

Koopman (1968) extended his original work to the two-sided search situation. It is shown that a uniform distribution

of search effort on the part of the searcher and a uniform position density on the part of the target form the optimal strategies for the resulting zero-sum game.

Neuts (1963) develops, among other things, stationary minimax strategies for a multistage search game. The objective is the minimization of the expected discounted return to the target. A stationary strategy for the hunter is an n-tuple

$$Y = (Y_1, \ldots, Y_N) \; ; \; Y_i \geq 0, \; \sum_{i=1}^{N} Y_i = 1 ,$$

which denotes a probability distribution, chosen once, and by which the region to be examined at each stage is selected. It is shown that the optimal stationary policy for the searcher is independent of all parameters except the discrete detection probabilities, a result which also holds for the objective of minimizing the expected duration of the game. Such stationary minimax strategies correspond to the following cases:

(a) a memoryless searcher,

(b) the target is allowed to move after each region is searched.

Charnes and Schroder (1967) develop models and methods to find optimal tactics in an idealization of antisubmarine warfare, viewed as a game of pursuit between the hunter force

and a submarine. The objective function of minimizing the expected duration of the search can be expressed as a stochastic game. The solution of this game is accomplished by solving a sequence of linear programming problems. In the event the hunter knows the behavior of the submarine, the game becomes a one-person game and may be treated as a discounted Markovian decision process of the type studied by Howard (1960).

1.2.3 Search and the Visibility Process

As noted earlier, Koopman (1946) briefly examined the situation in which the probability of target presence is modified by the inclusion of the probability that the target is visible given that it is present.

Stollmack (1968) determined by both field and laboratory experimentation that the exponential detection function with constant rates is a valid model of visual detection. The study centered on the visual detection of tanks by experienced personnel in the terrain surrounding Fort Knox, Kentucky. The detection rates obtained by Stollmack were shown to be statistically dependent upon range and background (i.e., the number of confusing forms, ruggedness of the terrain, etc.). The empirical relationships indicated that at mid to high ranges the detection rate is as sensitive to changes in the background as it is to changes in range. It was shown that differing detection rates and

visibility conditions were the rule rather than the exception over the local terrain.

Bonder (1970) and Disney (1970) formulated descriptive models of the situation in which the target visibility changes over time. Bonder considered the situation in which the target and the searcher (detector) may not be continuously visible during the period of time in which the searcher is examining the subregion containing the target. The searcher has a detection capability only when the target is visible.

The author considered the following situations within each search area:

(a) The target may be visible to the searcher for the entire search interval with some known probability p (this is the Koopman suggestion),

(b) the target may be visible at the start of the search period, the length of the visible period being a random variable with known probability density function, and not reappear,

(c) a single period of visibility may be exhibited starting at some random time during the search interval and lasting a random amount of time.

In each of these cases it is assumed that the target is stationary in the sense that it remains in a given region, although its motion within a region may give rise to the visibility process. These models are the first to interface the visibility and detection processes. In each of these cases, the probability density functions for the time until detection, the time spent searching the area until a fixed

number of detections occur, and the time spent searching the total area, including switching times, are derived.

Disney described the visibility process in which the target alternates between visible and invisible states as an alternating renewal process. He did not consider the interface between the visibility and the detection processes. The transition matrix for this process is

$$\begin{array}{ll} & \begin{array}{cc} \underline{\text{Vis.}} & \underline{\text{Masked}} \end{array} \\ \begin{array}{l} \text{Vis.} \\ \text{Masked} \end{array} & \begin{pmatrix} 0 & f_1(t) \\ f_2(t) & 0 \end{pmatrix} \end{array}$$

where $f_1(t)$ is the probability density function for the time in the visible state and $f_2(t)$ the probability density function for the time in the masked state.

Employing some renewal theory arguments, the author obtained among other things:

(a) $\pi_1(t)$, the probability that the target is visible in $(t,t + dt)$,

(b) for a fixed time interval of length τ_d, the distribution of:

  (1) the number of times the target is visible,

  (2) the total time of visibility.

1.2.4 *Summary and Conclusions from the Literature Review*

The review of the open literature on search and reconnaissance theory indicated that the bulk of the research activity has dealt with purely mathematical extensions of the

work of Koopman. In nearly every case, no attempt was made to relate the mathematical models to real world situations. It seems apparent that more effort should be directed toward the study of search models which afford more realistic descriptions of search scenarios.

A recent research report (Vector (1970)) has noted the importance of modeling the effects of terrain on combat processes. Since the reconnaissance activity is a prelude to such activities, one should also be concerned with the effects of terrain on the search activity, i.e., how does the inclusion of the visibility process effect the optimal search strategies and returns?

Realistically, the optimization criteria should depend upon the objective of the operation. Research should be devoted to the structuring of the total activity, which includes search, detection, tracking, and ensuing action, before selecting the optimization criteria.[1]

Research is required to understand the behavior of operationally useful devices, e.g., the effect of multiple scans, independence between successive looks, coupling of various types of detectors, etc. In general, a study of the structure and capabilities of operational detectors is required.

[1] The relationship between search and ensuing action is being studied under this contract and is being published as a separate report (Kronz, 1971).

The output of many of the "classical" search studies has been a fixed amount of time to spend searching a region. The likelihood that searchers will not (or cannot) follow optimal search procedures suggests research be devoted to the problem of converting theoretical results into practical rules of application.

The open literature on search and reconnaissance is virtually vacuous in the important areas of multiple detectors and/or multiple targets with varying degrees of dependency within each group. Clearly all of the topics outlined in the previous paragraphs are of interest when viewed in the context of multiple detectors and/or targets.

1.3 Research Area

This dissertation addresses the problem of characterizing the interaction between the detection and visibility processes. The work of Bonder (1970) and Disney (1970) is extended and the resultant models analyzed to determine strategies for the optimal allocation of search effort. Comparisons with existing search theory strategies are made to determine conditions under which different strategies, which consider the visiblity process effects, are required. The models are further analyzed to gain insight into the intelligence-gathering process.

Specifically, the problem considered is that of searching for a stationary target located in one of N regions (discrete search space) with prior probability vector P = $(p_1, \ldots, p_N)$. Analogous formulations for the continuous search space are discussed in an appendix. Although the solution procedures and (we believe) the results carry through for the N region case, explicit policies and returns are given only for the two-region situation to facilitate both the analysis and interpretation of results. We principally consider the case of a continuous detector. Discrete detector formulations and associated optimal policies also are contained in an appendix. Within the region in which the target is located, it is assumed that during a given period of time in which a search for the target is underway, the target may exhibit one of the following types of behavior as indicated in Bonder (1970) and Disney (1970).

(a) The target may be visible to the searcher for the entire interval with some fixed probability v. This case is also referred to as the "binary" intervisibility process in the following discussions.

(b) The target may be visible at the start of the search period, the length of the visible period being a random variable with known probability density function.

(c) A single period of visibility may be exhibited, starting at some random time during the search interval and lasting a random amount of time.

(d) During the searching interval, the target may exhibit alternating periods of visibility and invisibility, the durations of each being random variables.

Physically, these visibility structures can be considered in the following contexts:

(1) Search for a target assigned to one region in which the local terrain, foliage, weather, etc., contributes to masking effect.

(2) Search for a submarine in which environmental conditions between the surface searching vessel and the target generate the visibility periods, e.g., thermal barriers to sonar detection devices.

(3) In the case of a single interval of visibility, the random length of this period could be considered the length of time until the target discovers the presence of the searcher in his region.

(4) The multiple periods of visibility could also reflect the target's strategy with regard to exposing himself in order to obtain information, supplies, etc.

(5) The multiple periods of visibility might correspond to the periods a raiding party of guerillas, operating from a neutral or safe zone, spends in a region in which they are susceptible to detection.

(6) Searching for schools of fish which periodically submerge to depths which preclude their detection.

(7) The situation in which a data bank is simultaneously accessed by many users. When one user has access to certain information, the others are temporarily denied access.

(8) The problem of detecting a disease, the symptoms of which exhibit intermittent remission, may be of interest.

(9) The time until a crime, e.g., a robbery, may be considered a random variable, as well as the length of time required to carry out the act.

(10) The general formulation of the search problem can be interpreted readily in a research and development context, i.e., one interprets the probability of detection as the probability of "discovery." However, the single interval of visibility with a random start time lends itself to some interesting interpretations in the research and development context. In many research situations, one may have some random period of time, effort, etc., in which the probability of discovery is essentially zero. Of course, there may be some alternatives for which this "settling in" period is unnecessary. Given that one attains the end of the "settling in" period, one may then estimate, based perhaps on current market conditions, the probability density function for the time required to complete the job in order that the end product will be competive, timely, etc.

Although (d) essentially includes structures (a)-(c) of the visibility process, the descriptive models, associated allocation strategies, and analyses of the models are presented in order of increasing complexity for pedagogical reasons at the expense of increasing the quantity of

information, since the research was performed in this fashion.[1] Because of this writing approach, the principal new results in search theory are not encountered until the latter part of Chapter 3 and thereafter.

Although the current research was motivated by the desire to learn more about the effects of the visibility processes on reconnaissance in the military context, a number of non-military interpretations of the visibility processes may be made as well.

[1] Morris (1967), Bonder (1971), and Wagner (1971) have indicated that a principal capability needed, but usually not developed, in new analysts is the process of model development. Writing the paper in this fashion will indicate "how" it is done, in contrast to presentation of just the end product which typically appears in journal articles.

Chapter 2

THE BINARY VISIBILITY MODEL

The Binary Visibility model is described and analyzed in this chapter. It assumes that, upon the entry of the searcher into a region containing the target, the target is visible to the searcher for the duration of the search with some fixed probability which may be regionally dependent. The problem of maximizing the probability of detection under a constraint on the available search time is studied in detail. Optimal solutions to this problem are obtained for the situation in which the searcher utilizes a continuous detection device. Comparisons are then made between these optimal policies and those for which the Binary process is ignored (the "Koopman" models). A sensitivity analysis of both the optimal policy and its return is carried out. The objective of minimizing the expected time until detection is also examined.

2.1 Description

We consider only the discrete search space version of this problem throughout this research. The discrete search space version of the problem of maximizing the probability of detecting the target is

$$\max \sum_{i=1}^{N} p_i v_i P_i(t_i)$$

$$\text{S.T.} \sum_{i=1}^{N} t_i \leq T \quad , \qquad (1)$$

and

$$t_i \geq 0 \quad ,$$

where

p_i = probability that the target is in the i^{th} region,

v_i = probability that, in the i^{th} region, the target and the searcher are intervisible for the entire period,

$P_i(t)$ = the conditional probability of detecting a target in the i^{th} region at or before time t.

As noted in Chapter 1, Koopman (1946) gave a mathematical justification (The Law of Random Search) for a detection function of the form

$$P_i(t) = 1 - e^{-k_i t} \qquad (2)$$

where k is the conditional detection rate.

More recently, Stollmack (1968), studying visual detection, established the validity of the above expression via experimental data. Therefore, (2) is used as the standard form for a continuous detection device. Observe that (2) is a concave function of search time t. Hence, upon substitution of (2) into (1), one obtains a concave maximization problem.

2.2 *Allocation of Effort to Maximize the Probability of Detection*

2.2.1 *Model Solution*

In this section, we shall develop the optimal policy for allocating search time for the Binary model given by (1) and (2) of 2.1. The reader is referred to Appendix A for the proof of the modified Charner-Cooper Algorithm which can be used to determine the optimal allocation $(t_1, \ldots, t_N)$ for the N-region problem. We shall discuss only the attributes of the 2-region situation for the purposes of economy of notation and the inherent symmetry of the situation.

In Appendix A, we develop the rule for determining the order in which regions begin to receive search effort (First Allocation Rule [FAR]). For this model the FAR may be stated:

Choose the region j for which

$$p_j k_j v_j = \max_{1 \leq i \leq N} \{ p_i k_i v_i \} .$$

Equation 8 of Appendix A suggests that all effort be placed in region 1 (assuming $p_1 k_1 v_1 \geq p_2 k_2 v_2$) when $T < T^{**}$, where T is the total available search time and

$$T^{**} = \frac{1}{k_1} \ln \left(\frac{p_1 k_1 v_1}{p_2 k_2 v_2} \right) . \tag{3}$$

For $T \geq T^{**}$, and using an analog of (9) in Appendix A, one obtains the following for the optimal allocations to the two regions,

$$t_1 = \frac{1}{k_1 + k_2}\left\{\ln\left(\frac{p_1 k_1 v_1}{p_2 k_2 v_2}\right) + k_2 T\right\},$$
$$t_2 = \frac{1}{k_1 + k_2}\left\{\ln\left(\frac{p_2 k_2 v_2}{p_1 k_1 v_1}\right) + k_1 T\right\}. \tag{4}$$

The optimal value for the probability of detection is then given by[1]

$$P(T) = p_1 v_1\left[1 - \left[\frac{p_1 v_1 k_1}{p_2 v_2 k_2}\right]^{-\frac{k_1}{k_1+k_2}} e^{-\left(\frac{k_1 k_2}{k_1+k_2}\right)T}\right] + p_2 v_2\left[1 - \left[\frac{p_2 v_2 k_2}{p_1 v_1 k_1}\right]^{-\frac{k_2}{k_1+k_2}} e^{-\left(\frac{k_1 k_2}{k_1+k_2}\right)T}\right]. \tag{5}$$

2.2.2 Comparison with the Koopman Model

In this section, we examine the situation in which a searcher, being aware of the earlier results of Koopman (which assume continuously visible targets), applies them to situations in which the target is not continuously visible because he is (a) aware of the visibility process but unwilling or unable to obtain estimates of the visibility parameters, or (b) unaware of the visibility process. We utilize the classical Koopman model since most efforts in search

[1]Formulations and results for discrete detections are given in Appendix B. Continuous search space versions of the model are discussed in Appendix C.

theory have been essentially embellishments of it and such comparisons will provide landmarks regarding the robustness of the classical models as guides for search decision makers. The following paragraphs, which form a conceptual summarization of the searcher's position in this respect as well as the options available to him, are introduced only for illustrative purposes.

First we assume that the searcher can obtain estimates of the visibility parameters at some additional cost (e.e., weather studies, terrain analysis, etc.). In this situation, the searcher may take the following actions:

(a) Allocate the search effort according to the Koopman scheme (without obtaining any estimates on the visibility parameters).

(b) Spend additional funds to obtain the required estimates on the visibility parameters).[1]

(c) Expend additional search effort under the Koopman policy in an attempt to make up for the defects resulting from using the "Koopman" policy.[2]

---

[1] One might assume that various levels of spending yield varying degrees of accuracy in the parameter estimates. In this situation, it is of interest to determine the sensitivity of the optimal allocations and returns to changes in the visibility parameters.

[2] This will be shown to be a meaningful alternative.

The outcomes of such actions are, respectively:

(a) A difference in the probability of detecting the target under a constraint on the total search time resulting from the use of the nonoptimal policy, denoted as ΔP.

(b) By obtaining the estimates of the visibility parameters, he can apply the optimal policy. A measure of the effectiveness of this option is given by $\Delta P/C_E$, where C_E is the cost of obtaining the estimates of the visibility parameters.

(c) The additional search effort needed to achieve the optimal return has an associated cost C_T. The gain from such an action is computed from the function which expresses the difference in the optimal return under the two policies, E(T), as a function of the total available search effort, T. Note the gain from this action will be computed from the changes in return under the additional effort allocated via the Koopman policy (we label this gain $\Delta P(T)$); however, the costs, C_T, may be less than C_E. A measure of the effectiveness of this action is given by $\Delta P(T)/C_T$.

In the situation in which the searcher is unable to obtain estimates of the visibility parameters or is unaware of the visibility process, the searcher may take the following actions:

(a) (Unaware) Allocate the search effort according to the Koopman scheme.

(b) (Unable) Expend additional search effort.

The consequences of such actions are described by outcomes (a) and (c), respectively, of the first case.

The solution for the Koopman problem is given by (4), with $v_i = 1$, for every i. One also observes that in these equations if $v_1 = v_2 \neq 1$, then the optimal allocation for the Binary model is identical to that of the Standard Koopman Allocation (SKA); hence, no error results from the use of the SKA in the Binary model. (The result is also true in the n-region situation, see Appendix A.)

Since we have closed-form expressions for both allocation procedures, they can be substituted into the objective function, to form the difference function

$$E(T) = P_B(T) - P_{SKA}(T). \tag{6}$$

$$E(T) = e^{-\frac{k_1 k_2}{k_1+k_2} T} \left\{ p_1 v_1 \left(\frac{p_1 k_1}{p_2 k_2}\right)^{-\frac{k_1}{k_1+k_2}} \left[1 - \left(\frac{v_1}{v_2}\right)^{-\frac{k_1}{k_1+k_2}}\right] + p_2 v_2 \left(\frac{p_2 k_2}{p_1 k_1}\right)^{-\frac{k_2}{k_1+k_2}} \left[1 - \left(\frac{v_2}{v_1}\right)^{-\frac{k_2}{k_1+k_2}}\right]\right\}.$$

We note that $\lim_{T\to\infty} [P(T) - P_{SKA}(T)] \to 0$. Equation 6 only holds where, under both policies, both regions are receiving positive allocations of search effort. In the event this situation doesn't hold, one has four cases to consider.

The switch points (the values of total available search effort, T, at which the second region begins to receive some effort) are given by:

(a) for SKA

$$T^* = \frac{1}{k_1} \ln \frac{p_1 k_1}{p_2 k_2}$$

(b) for the Binary model [7]

$$T^{**} = \frac{1}{k_1} \ln \frac{p_1 k_1}{p_2 k_2} + \frac{1}{k_1} \ln \frac{v_1}{v_2} .$$

If it is assumed that Region 1 is selected first by both policies, the following situations can arise as a result of the difference in switch points:

(a) $T < T^* \leq T^{**}$, the error is zero, since under both policies all the effort is placed in Region 1.

(b) $T^* < T < T^{**}$, under SKA the effort is divided according to equation 4, while under the Binary allocation all effort is still placed in Region 1, resulting a nonzero error.

(c) One could have $v_1 < v_2$, in which case $T^{**} < T < T^*$, and the reverse of (a) and (b) would occur, again the error term is nonzero.

(d) $T^* < T^{**} < T$, one obtains equation 6 as the error expression.

Suppose $p_1k_1 > p_2k_2$, but $p_1k_1v_1 < p_2k_2v_2$, i.e., the infinitesimal detection rate is largest in the first region; however, the probability that the target is visible is much higher in the second region. In this situation the SKA selects Region 1 and allocates all effort to that region until $T > T^*$, while the optimal policy for the Binary model selects Region 2 and allocates all effort (T) to that region until $T > T^{**}$. In Figures 1 and 2 we illustrate the above points, by plotting the percent relative error, $E(T)/P_B(T)$, in the probability of detection using the Koopman policy versus the total available search time. Figure 1 contains examples where both policies begin searching the same region. One may note that the peak percent relative error appears to occur at the switch point for the Koopman model, T^*. The peak error functions and their times of occurrence will be discussed later in this section.

Figure 2 contains examples in which the Koopman policy results in the wrong region being searched. These are situations in which the total available search is highly constrained. This suggests that, in such situations, the decision maker must be aware of the visibility probabilities for the search space in order to conduct an effective search.

An analysis of the error function, $E(T)$, was undertaken to determine:

(a) At what levels of total available time (T) the peak differences occur.

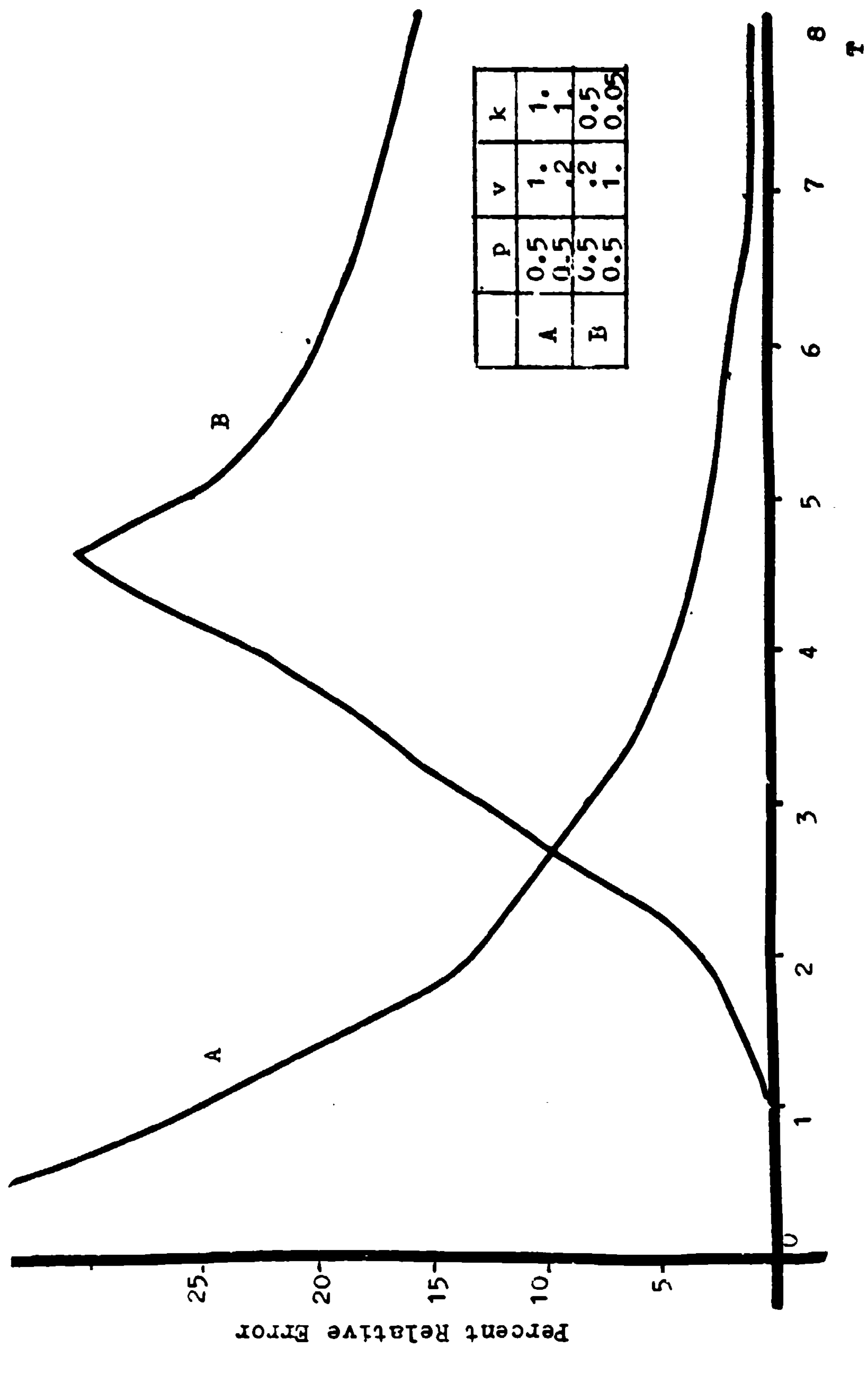


Figure 1 - Percent Relative Error Versus Search Time

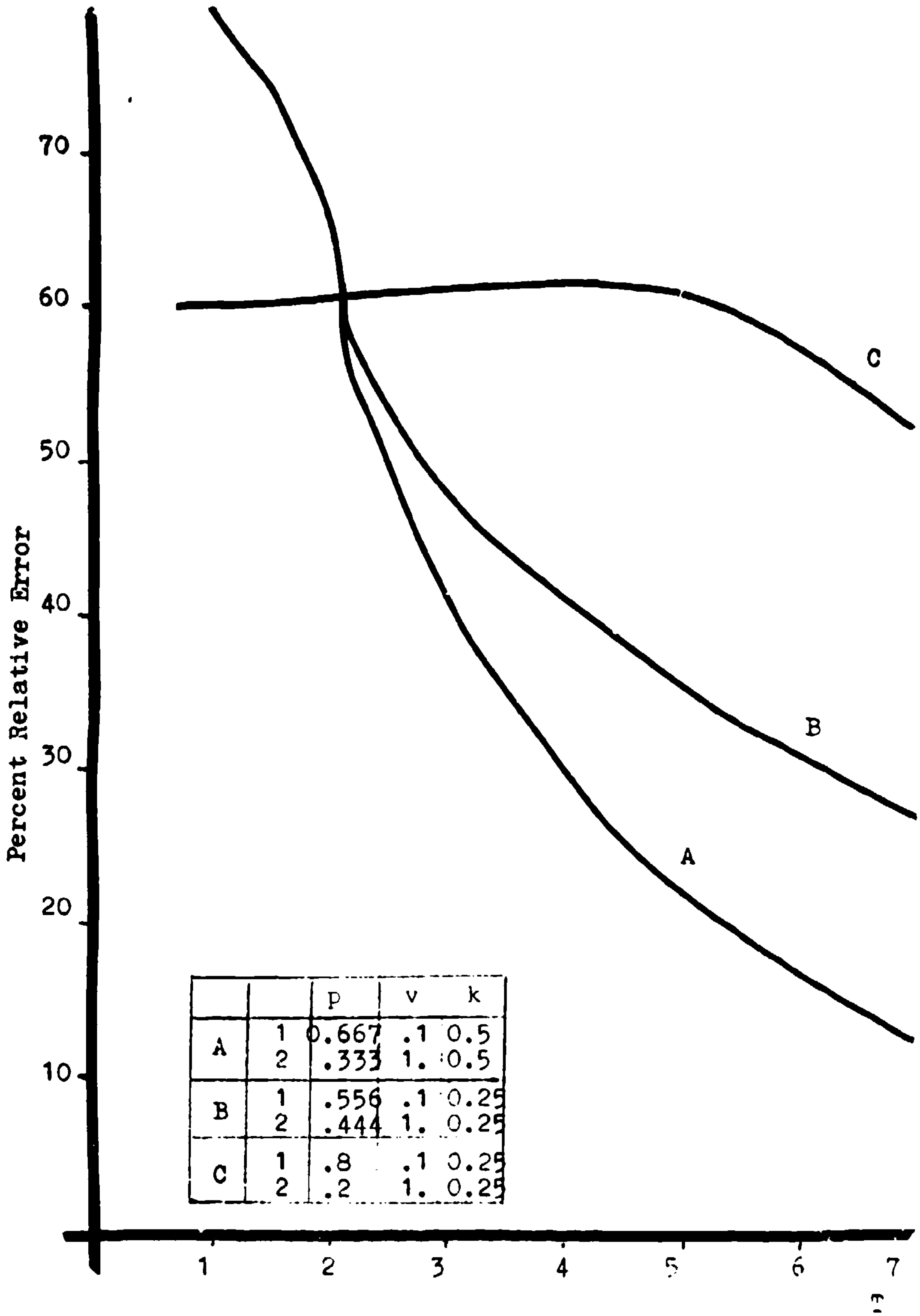


Figure 2 - Sensitivity of the Error Function

(b) What determines their shape, i.e., height, rate of decay, etc.

For the two-region Binary Search Model, there are essentially four sets of parametric relationships, since one need only interchange the labels to obtain the others. These relationships are outlined below.

For the SKA we assume that

$$\frac{p_1}{p_2} \geq \frac{\frac{1}{k_1}}{\frac{1}{k_2}} \quad ,$$

Case I: $p_1k_1v_1 \geq p_2k_2v_2$, $(v_1/v_2) \geq 1$, and $T^{**} \geq T^*$

Case II: $p_1k_1v_1 \geq p_2k_2v_2$, $(v_1/v_2) < 1$, and, $T^* > T^{**}$

CaseIII: $p_1k_1v_1 \leq p_2k_2v_2$, $(v_1/v_2) < 1$, and $T^* \leq T^{**}$

Case IV: $p_1k_1v_1 \leq p_2k_2v_2$, $(v_1/v_2) < 1$, and $T^* > T^{**}$.

Expressions for the peak error functions and the level of search time at which the peaks occur are developed and analyzed below.

<u>Case I</u>

In the first situation, the peak difference occurs at

$$T_p = T^* - \frac{(k_1 + k_2)}{k_1^2} \ln\left[\frac{k_2}{k_1 + k_2} + \left(\frac{v_2}{v_1}\right)\frac{k_1}{k_1 + k_2}\right],$$

substituting this expression into the error function gives

$$E(T_p) = p_1 v_1 \left(\frac{p_1 k_1}{p_2 k_2}\right)^{-1} \left[1 - \left(\frac{k_2}{k_1 + k_2} + \left(\frac{v_2}{v_1}\right) \frac{k_1}{k_1 + k_2}\right)\right]\left(\frac{k_2}{k_1 + k_2}\right.$$

$$\left. + \left(\frac{v_2}{v_1}\right) \frac{k_1}{k_1 + k_2}\right)^{k_2/k_1} - p_2 v_2 \left[1 - \left(\frac{k_2}{k_1 + k_2}\right.\right.$$

$$\left.\left. + \left(\frac{v_2}{v_1}\right) \frac{k_1}{k_1 + k_2}\right)^{k_2/k_1}\right].$$

Case II and IV yield the following results

$$T_p = T^* = \frac{1}{k_1} \ln \left(\frac{p_1 k_1}{p_2 k_2}\right),$$

and substitution into the error function gives

$$E(T_p) = p_2 v_1 \left(\frac{k_2}{k_1}\right)\left[1 - \left(\frac{v_2}{v_1}\right)^{\frac{k_1}{k_1+k_2}}\right] + p_2 v_2 \left[1 - \left(\frac{v_2}{v_1}\right)^{-\frac{k_2}{k_1+k_2}}\right]$$

The peak error is identical for Cases II and IV; however, the shapes of the error functions are different since in Case IV the Koopman policy starts in the wrong region, while in Case II both policies start in the same region.

Case III

In the third case, some subcases must be examined:

for

$$\left(\frac{k_2}{k_1} + 1\right) \ln\left(\frac{p_1 k_1}{p_2 k_2}\right) < -\left(1 + \frac{k_1}{k_2}\right) \ln\left[\frac{k_1}{k_1 + k_2} + \left(\frac{v_1}{v_2}\right) \frac{k_2}{k_1 + k_2}\right]$$

$$T_p = \frac{1}{k_2} \ln\left(\frac{p_2 k_2}{p_1 k_1}\right) - \frac{k_1 + k_2}{k_2^2} \ln\left[\frac{k_1}{k_1 + k_2} + \left(\frac{v_1}{v_2}\right) \frac{k_2}{k_1 + k_2}\right]$$

and

$$E(T_p) = v_2 p_1 \frac{k_1}{k_2} \left[\frac{k_1}{k_1 + k_2} + \left(\frac{v_1}{v_2}\right) \frac{k_2}{k_1 + k_2}\right]^{k_1/k_2}$$

$$\cdot \left\{1 - \left[\frac{k_1}{k_1 + k_2} + \left(\frac{v_1}{v_2}\right) \frac{k_2}{k_1 + k_2}\right]\right\}$$

$$- p_1 v_1 \left\{1 - \left[\frac{k_1}{k_1 + k_2} + \left(\frac{v_1}{v_2}\right) \frac{k_2}{k_1 + k_2}\right]^{k_1/k_2}\right\},$$

In the event this condition doesn't hold,

$$T_p = T^* = \frac{1}{k_1} \ln\left(\frac{p_1 k_1}{p_2 k_2}\right),$$

and

$$E(T_p) = p_2 v_2 \left[1 - \left(\frac{p_1 k_1}{p_2 k_2}\right)^{-k_2/k_1}\right] - p_1 v_1 \left[1 - \left(\frac{p_1 k_1}{p_2 k_2}\right)^{-1}\right].$$

It can also be shown that the peak error, at least in the two-region case, will always occur at or before the larger of (T^*, T^{**}) as follows. One need only observe that for $T > \max(T^*, T^{**})$ the error function given by equation 6, is a decreasing function of the total available effort T. This implies that in general the maximum error appears in situations in which the search effort is highly constrained.

Having closed-form expressions for the peak error functions, one can conceptually take partial derivatives to determine the sensitivity of these functions to changes in (p_i, v_i, k_i). However, this is algebraically messy. A straightforward numerical search for extrema over the entire search parameter space was undertaken and produced the results summarized in Table 1. From Table 1, it is apparent that the critical factor is the ratio of v_1/v_2, e.g., for areas having homogeneous visibility (equal visibility parameter values) the peak differences are small, suggesting the adequacy of the Koopman policy. Note that, in general, one obtains greater errors whenever the ratios of the prior probabilities and the visibility probabilities are opposite in magnitude, suggesting, in the case $k_1 \approx k_2$, that the peak error results from an erroneous selection of the initial region searched.

A secondary analysis produced differences in the probability of detection in the range (.12, .17) for the following

Table 1

Maximum Differences in the Probability of Detection

$p_1/p_2 = .1$

k_2/k_1 \ v_1/v_2	.1	1.	10.	100.
.1	.027	0.	.045	.063

$p_1/p_2 = 1.$

k_2/k_1 \ v_1/v_2	.1	1.	10.	100.
.1	.059	0.	.025	.035
1.	.101	0.	.10	.1225

$p_1/p_2 = 10.$

k_2/k_1 \ v_1/v_2	.1	1.	10.	100
.1	.0106	0.	.0048	.006
1.	.0425	0.	.018	.0225
10.	.048	0.	.027	.031

$p_1/p_2 = 100.$

k_2/k_1 \ v_1/v_2	.1	1.	10.	100.
.1	.0118	0.	.0005	.0007
1.	.0468	0.	.002	.0024
10.	.0644	0.	.003	.0034
100.	.065	0.	.003	.0036

values of the search parameters

$$1 \leq p_1/p_2 \leq 3, \ .01 \leq v_1/v_2 \leq .1, \ .5 \leq k_1/k_2 \leq 1.5 \ .$$

Finally, we note that the above analysis deals only with the peak differences and yields no information about the rate of decay of the error function with increasing available effort.

Table 2 presents some additional measures of the adequacy of using the Koopman policy when in fact the target is not continuously visible. The measures presented in Table 2 consist of:

(a) The expected time to detect using the Koopman strategy and model to allocate an unlimited amount of effort, labeled KET (Appendix A). It is conjectured that a searcher unaware of the visibility process may use such a measure to characterize a given search situation.

(b) Since in the Binary Model the ultimate probability of detection is less than unity, one can only make comparisons on the basis of the *conditional* expected times to detect, conditioned on detection occurring. Table 1 contains the ratio of these conditional expected times until detection. The numerator is computed using the Koopman strategy in the Binary model situation while the denominator is computed from the optimal policy for the Binary Model. (See 2.3.)

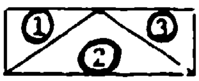
(c) Given that one is using the Koopman strategy in the context of a Binary search situation, it is of interest to determine how much search time must be

Table 2

Binary Model Sensitivity

p_1/p_2	k_2/k_1	v_1/v_2 = 0.05 (1)	0.05 (2)	0.05 (3)	v_1/v_2 = 0.1 (1)	0.1 (2)	0.1 (3)
1	0.5	1.42	.04 4.	1.53	1.42	.03 4.	1.38
	1.0	2.	.049 5.	1.62	2.	.037 2.	1.44
	1.5	1.64	.03 5.	1.59	1.64	.042 4.	1.41
2	0.5	1.23	.025 5.	1.61	1.23	.035 4.	1.39
	1.0	1.9	.04 5.	1.86	1.9	.05 4.	1.56
	1.5	1.65	.036 5.	1.95	1.65	.05 4.	1.60
3	0.5	1.1	.028 5.	1.6	1.1	.037 4.	1.36
	1.0	1.78	.047 6.	1.9	1.78	.032 6.	1.55
	1.5	1.6	.045 5.	2.05	1.6	.031 5.	1.63

1. KET-Koopman Expected Time.



2. E(T)/P(T): Value of E(T)/P(T) and time at which E(T)/P(T) < .05.
3. Ratio of the conditional expected times until detection under the respective policies.

available in order that the ratio of the difference function to the probability of detection under the optimal policy be less than some specified value. Assuming a level of 0.05, the table gives the total search time required to reach that level using the Koopman scheme.

Thus, it would appear, from Table 2, that for situations in which t' searcher (decision maker) has large quantities of search time, (T > 2 KET), he need not be concerned with having good estimates of the visibility parameters. Qualitatively, equation 6 enables one to answer such questions over a range of possbile values for the visibility parameters. For example, assume one wishes to determine that value of T, the total available time for which the percent relative error is below some preassigned level, say P_0. By equating the error function, equation 6, to this value and solving for T, one obtains the required level of time. Table 2 contains some numerical analyses of this nature. For example, in the situation in which $p_1/p_2 = 3$ and $k_1/k_2 = 1$, if the decision maker estimates the ratio of v_1/v_2 at 0.1 and assigns 6 units of search time, such an assignment will yield a relative error of less than 0.05 even if v_1/v_2 turned out to be as low as 0.05. Of course, if the ratio turned out to in the interval [.1,1], the actual percent relative error will be even less. Note also from Table 2 that the 6 units of search time are approximately three times the Koopman expected time until detection. The searcher has the choice

of obtaining estimates of v_1 and v_2 at some cost (e.g., weather information), or expending extra effort by disregarding the visibility process and following the Koopman scheme. If the decision maker has means of obtaining estimates of v_1 and v_2, the above analysis provides a sensitivity study of the cost to him of using the Koopman scheme, i.e., failing to obtain them.

2.2.3 Model Sensitivity

In the following paragraphs, we present the results of a sensitivity analysis with respect to the model visibility parameters for both the optimal policy and its associated return. In order that the following results hold, it is necessary to assume that the available search time is larger than T**.

Consider first, the sensitivity of the optimal policy. The relevant partial derivatives are listed below. For the optimal allocation to region 1, one has

$$\frac{\partial t_1}{\partial v_1} = \frac{1}{k_1 + k_2}\left(\frac{1}{v_1}\right) \text{ and } \frac{\partial t_1}{\partial v_2} = \frac{-1}{k_1 + k_2}\left(\frac{1}{v_2}\right).$$

For region 2, we have

$$\frac{\partial t_2}{\partial v_2} = \frac{1}{k_1 + k_2}\left(\frac{1}{v_2}\right) \text{ and } \frac{\partial t_2}{\partial v_1} = \frac{-1}{k_1 + k_2}\left(\frac{1}{v_1}\right).$$

First, some general observations. Note that the larger v_i $(0 < v_i \leq 1)$, the less sensitive the optimal allocation to changes in v_i. Also, the optimal policy is fairly robust

with respect to changes in the visibility parameters whenever the detection rates k_1, k_2 are large. Finally, we note that the above partials are independent of the level of available search effort. This suggests that the optimal policy will be much more sensitive to changes in the visibility parameters whenever the total search effort is limited. We note in advance that such independence will not be the case for the models of Chapters 3 and 4. These results imply that:

(a) the decision maker need not be concerned about small errors in his estimates of the v_i in the situation in which the detection rates are high, and

(b) as the estimates of v_i tend toward unity, small errors in them will have little effect on the optimal policy unless the detection rates are very small.

Next we examine the sensitivity of the optimal return function to changes in the visibility parameters. In order that results remain valid, the available search time must be larger than T**. The partial derivatives are listed below with comments whenever appropriate:

$$\frac{\partial P(T)}{\partial v_1} = p_1 \left[1 - e^{-\frac{k_1 k_2}{k_1+k_2} T} \left(\frac{p_1 k_1 v_1}{p_2 k_2 v_2} \right)^{-\frac{k_1}{k_1+k_2}} \right] \geq 0 \ ,$$

$$\frac{\partial P(T)}{\partial v_2} = p_2 \left[1 - e^{-\frac{k_1 k_2}{k_1+k_2} T} \left(\frac{p_2 k_2 v_2}{p_1 k_1 v_1} \right)^{-\frac{k_2}{k_1+k_2}} \right] \geq 0 \ .$$

Here we note that the sensitivity of the optimal return increases with increasing search time, T. However, the maximum sensitivity is determined by the prior probabilities on target presence. The greater the prior probability of target presence in a region, the more sensitive the optimal return to changes (errors) in the estimates of the visibility parameters. The above interpretation doesn't imply that the optimal policy is robust for small values of total available search time, since, on a relative basis, relative to the optimal return, the sensitivity is higher for small values of search time.

We can summarize our analysis of the Binary process by noting that the important effects of this type of process occur under nonhomogeneous visibility conditions and for search situations with limited total available search time.

2.3 *Minimization of the Expected Time Until Detection*

2.3.1 *Model Solution*

Recall the arguments given in Appendix A for the value of the objective function in problem of minimizing the expected time to detect. These will hold here also with the following important difference. The optimal value of the probability of detection for the Binary model, under a constraint on the total search time, is in the limit as $T \to \infty$.

$$P(\infty) = p_1 v_1 + p_2 v_2 < 1, \quad \text{for } 0 < v_1, v_2 < 1 .$$

Then with probability $(1 - P(\infty))$ the search is unsuccessful, and the expected time to detect increases without limit. In order to see this more clearly, consider the following expression for the expected time to detect:

$$E = \lim_{T \to \infty} \left\{ \int_0^T t dP(t) + T(1 - P(T)) \right\} .$$

For the Binary model

$$\lim_{T \to \infty} (1 - P(T)) \to \varepsilon > 0 ,$$

hence, the unbounded expected time to detect.

One can, however, consider the conditional expected time to detect, given that detection ultimately occurs, i.e.,

$$E = \int_0^\infty \frac{t dP(t)}{P(\infty)} .$$

Proceeding as in Appendix A under the assumption that the searcher starts in region 1 , we obtain for the optimal value of this objective function.

$$E = \frac{1}{P(\infty)} \left[p_1 v_1 \left[\frac{1}{k_1} \left(1 - e^{-k_1 T^{**}} \right) + \left[\frac{p_1 k_1 v_1}{p_2 k_2 v_2} \right]^{-\frac{k_1}{k_1 + k_2}} \left\{ 1 + \frac{k_1}{k_2} \right\} e^{-\frac{k_1 k_2}{k_1 + k_2} T^{**}} \right] + p_2 v_2 T^{**} \right] ,$$

where T^{**} is given by (3).

For the case in which the infinitesimal detection rates are identical, i.e., $p_1 v_1 k_1 = p_2 v_2 k_2$, one has that

$$E = \frac{k_1 + k_2}{k_1 k_2} ,$$

which is identical to the unconditional expected time to detect for the standard model. Of course, E is the *conditional* expected time to detect; with probability $(1 - \dot{P}(\infty))$ the target is not detected.

2.3.2 *Comparison with the Koopman Model*

In this section we shall again consider the case of the partially informed searcher; however, the objective of the search will be the minimization of the conditional expected time until detection. Recall that our mode of comparison is that of a searcher who is aware of the Koopman (1946) results but is not aware of the target behavior and/or visibility conditions. In such a situation, he would make use of Dobbie's (1962) results for the expected time problem, i.e., the policy that maximizes the probability of detection is also the one which minimizes the expected time until detection, if the conditional detection functions are concave. Proceeding as in Appendix A, using the Koopman allocation policy in the Binary model, one obtains for the optimal value of the

conditional expected time until detection for a 2-region search situation,

$$E\cdot(p_1v_1 + p_2v_2) = p_1v_1\left\{\frac{1}{k_1}\left[1 - e^{-k_1T^*}\right] - Te^{-k_1T^*}\right\}$$
$$+ \left[p_1v_1\left(\frac{p_1k_1}{p_2k_2}\right)^{-p_1/p_1+p_2} + p_2v_2\cdot\right.$$
$$\left.\left(\frac{p_2k_2}{p_1k_1}\right)^{-k_2/k_1+k_2}\right]\cdot\left[T^*e^{-\frac{k_1k_2}{k_1+k_2}T^*}\right.$$
$$\left.+ \frac{k_1 + k_2}{k_1k_2}\, e^{-\frac{k_1k_2T^*}{k_1+k_2}}\right],$$

where $T^* = \frac{1}{k_1}\ln\left(\frac{p_1k_1}{p_2k_2}\right)$.

If region 2 is searched first, then T^* is defined accordingly and one makes the obvious changes in the first term of the above expression.

In order to complete our comparison with the Koopman model, we form the ratio of the measure just obtained to that resulting from using the optimal policy for the Binary model in the computation of the expected time until detection. Table 2 (repeated here for convenience) contains the results of several such comparison. It is interesting to note that:

(a) Although in certain cases the Koopman policy yields a small maximum difference in the probability of

Table 2
Binary Model Sensitivity

p_1/p_2	k_2/k_1	v_1/v_2 = 0.05 ①	0.05 ②	0.05 ③	v_1/v_2 = 0.1 ①	0.1 ②	0.1 ③
1	0.5	1.42	.04 4.	1.53	1.42	.03 4.	1.38
	1.0	2.	.049 5.	1.62	2.	.037 2.	1.44
	1.5	1.64	.03 5.	1.59	1.64	.042 4.	1.41
2	0.5	1.23	.025 5.	1.61	1.23	.035 4.	1.39
	1.0	1.9	.04 5.	1.86	1.9	.05 4.	1.56
	1.5	1.65	.036 5.	1.95	1.65	.05 4.	1.60
3	0.5	1.1	.028 5.	1.6	1.1	.037 4.	1.36
	1.0	1.78	.047 6.	1.9	1.78	.032 6.	1.55
	1.5	1.6	.045 5.	2.05	1.6	.031 5.	1.63

1. KET-Koopman Expected Time. ① ② ③

2. E(T)/P(T): Value of E(T)/P(T) and time at which E(T)/P(T) < .05.

3. Ratio of the conditional expected times until detection under the respective policies.

detection, the same policy yields an increase in excess of 50 percent in the conditional expected time until detection. For the Binary model, the long-term allocation rates are identical to those resulting from the Koopman policy. The larger ratios of the expected time in Table 2 occur, for the most part, as a result of the Koopman policy selecting the wrong initial region.

(b) The allocation of search time on the order of $\sim$3 KET, reduces the error function to less than 0.05; however, cost-wise, in terms of the minimum cost of searching, the Koopman policy is extremely inefficient for the ranges of Table 2.

(c) For the parameter ranges of Table 2, and a constant value of p_1/p_2, the ratio of the conditional expected times until detection tends to increase with increasing k_2/k_1. The effect of increasing k_2/k_1 is greater than that of increasing the ratio of prior probabilities.

Chapter 3

A RANDOM INTERVAL OF VISIBILITY: INITIALLY VISIBLE

The "single-interval-of-visibility" model is described and analyzed in this chapter. It is assumed that the target is visible upon the searcher's entry into the appropriate region; however, the length of time it remains visible is a random variable, the distribution of which is possibly regionally dependent. This model might describe a situation in which a target, operating in the area about to be searched, possesses detection gear. The length of the "visible" period is analogous to the length of time required for the target to become aware of the searcher's presence in his region.

The problem of maximizing the probability of detection under a constraint on the available search time is examined in detail. A comparison of both optimal policies and associated returns for the standard Koopman and random interval models is made. First-order sensitivity analyses on the optimal policy and return for this model are then described. The minimization of the conditional expected time to detect, given unlimited search effort, is also considered.

3.1 *Description*

As Bonder (1970) points out, the probability of detection in a given region under this model, in an infinitesimal interval $d\tau$, is given by

$$P(\tau)d\tau = Pr\{\tau \leq \tau_d \leq \tau + d\tau | \text{target presence } \& \; t_v > \tau\} Pr\{t_v > \tau\} \; ,$$

$$= \rho(\tau)d\tau \int_{\tau}^{\infty} h(t)dt \; ,$$

where

τ_d = time until detection,

$\rho(\tau)$ = probability density function for τ_d,

t_v = time that the target remains visible, and

$h(\tau)d\tau = Pr\{\tau \leq t_v \leq \tau + d\tau\}$.

Then the probability of detection in the i^{th} region given t, the time spent searching, is

$$P_i(t) = \int_0^t p_i \rho(\tau)\bar{H}(\tau)d\tau \; , \tag{8}$$

where

$$\bar{H}(\tau) = \int_{\tau}^{\infty} h(t)dt \ .$$

The problem of maximizing the probability of detection over a discrete search space is then given by

$$\max \sum_{i=1}^{N} p_i \int_0^{t_i} \rho(\tau)\bar{H}(\tau)d\tau$$

$$\text{S.T.} \quad \sum_{i=1}^{N} t_i \leq T \ , \qquad (9)$$

$$t_i \geq 0 \ .$$

3.2 Allocation of Search Time to Maximize the Probability of Detection

3.2.1 Model Solution

Recall the comments made in Chapter 2 concerning the theoretical and empirical relevance of the form of the conditional detection function we have assumed, i.e.,

$$\rho_i(\tau) = k_i e^{-k_i \tau} \ , \quad \tau \geq 0.$$

In view of the interpretation of the visible period as the length of time required for the target to detect the searcher, it is reasonable to assume that tne target has similar capabilities[1], i.e.,

[1]Additionally, personal correspondence between Professor Bonder and the staff of the Defense Operations Analysis Establishment indicate that visibility periods in terrain examined by the British do follow an exponential law.

$$h_i(\tau) = \lambda_i e^{-\lambda_i \tau} , \quad \tau \geq 0 .$$

Substitution into equation 8 yields

$$P_i(T) = \frac{k_i}{\lambda_i + k_i} (1 - e^{-(\lambda_i + k_i)T}) ,$$

and the maximization problem of (9) becomes

$$\max \sum_{i=1}^{N} p_i \frac{k_i}{\lambda_i + k_i} (1 - e^{-(\lambda_i + k_i)t_i})$$

$$\text{S.T.} \sum_{i=1}^{N} t_i \leq T , \qquad (10)$$

$$t_i \leq 0 ,$$

which is a concave maximization problem. The modified Charnes-Cooper algorithm given in Appendix A is again applicable. We again restrict our discussions to two regions for the reasons already noted. The solution to the N-region problem is obtained by following the procedure given in Appendix A.

The condition on available search time which places all the search time in region 1 is $T < T^{**}$,

$$\text{where} \quad T^{**} = \frac{1}{k_1 + \lambda_1} \ln \frac{p_1 k_1}{p_2 k_2} . \qquad (11)$$

We have assumed that region 1 is selected by the First Allocation Rule (Appendix A), i.e., $p_1 k_1 \geq p_2 k_2$. The First

Allocation Rule for this model is identical to that of the Koopman model.

Choose that region j for which $p_j k_j = \max_{1 \le i \le N} p_i k_i$.

Using an analog of equation 9, Appendix A, for the case $T \ge T^{**}$, one obtains, for the optimal allocations,

$$t_1 = \frac{1}{\lambda_1 + k_1 + \lambda_2 + k_2} \left\{ \ln\left(\frac{p_1 k_1}{p_2 k_2}\right) + (\lambda_2 + k_2)T \right\},$$

and (12)

$$t_2 = \frac{1}{\lambda_1 + k_1 + \lambda_2 + k_2} \left\{ \ln\left(\frac{p_2 k_2}{p_1 k_1}\right) + (\lambda_1 + k_1)T \right\}.$$

Note that in the limit as λ_1 & $\lambda_2 \to 0$ (the mean length of the visible period increases without limit) one obtains the Koopman model. The probability of detection under the optimal policy is then given by[1]

$$P(T) = \frac{p_1 k_1}{k_1 + \lambda_1} \left[1 - \left[\frac{p_1 k_1}{p_2 k_2}\right]^{-\frac{(\lambda_1 + k_1)}{\lambda_1 + k_1 + \lambda_2 + k_2}} e^{-\frac{(\lambda_1 + k_1)(\lambda_2 + k_2)}{\lambda_1 + k_1 + \lambda_2 + k_2} T} \right]$$

$$+ \frac{p_2 k_2}{k_2 + \lambda_2} \left[1 - \left[\frac{p_2 k_2}{p_1 k_1}\right]^{-\frac{(\lambda_1 + k_2)}{\lambda_1 + k_1 + \lambda_2 + k_2}} e^{-\frac{(\lambda_1 + k_1)(\lambda_2 + k_2)}{\lambda_1 + k_1 + \lambda_2 + k_2} T} \right].$$

(13)

[1]Formulations and results for discrete detectors are given in Appendix B. Continuous search space (continuous detector) versions of the model are discussed in Appendix C.

3.2.2 *Comparison with the Koopman Model*

In this section we examine the situation in which a partially informed searcher being aware of the earlier results of Koopman (which assume continuously visible targets), applies them to situations in which the target behavior is actually characterized by the random interval of visibility process.[1] The consequences (in terms of the probability of detecting the target under a constraint on the total available search time) of this erroneous application of the Koopman results will be compared to those obtained from the optimal allocations for these models.

First, we note that in equations 12 as $\lambda_1, \lambda_2 \to 0$, the optimal allocation policy becomes equivalent to the standard Koopman allocation (SKA). Consider the following comparisons between the optimal allocations of (12) and the SKA: for $p_1k_1 = p_2k_2$, and $k_1 = k_2 = k$, $\lambda_1 = \lambda_2 = \lambda$, the optimal allocations are identical and no error results from using the SKA. When $p_1k_1 = p_2k_2$, and $k_1 = k_2 = \lambda_1 = \lambda_2 = \gamma$, the same conclusion holds. These results also hold for n-regions

[1]The reader is referred to Section 2.2.3 for a detailed discussion of the implications of such an analysis to the decision maker (searcher).

(See Appendix A).[1] As shown earlier, the switch points are given by (assuming $p_1k_1 \leq p_2k_2$):

(a) for SKA

$$T^* = \frac{1}{k_1} \ln\left(\frac{p_1k_1}{p_2k_2}\right),$$

(b) for the single interval model

$$T^{**} = \frac{1}{k_1 + \lambda_1} \ln\left(\frac{p_1k_1}{p_2k_2}\right).$$

We observe that $T^{**} \leq T^*$ for $\lambda_1 > 0$. Intuitively, this is reasonable since the target will not remain visible forever in the first region, hence one must switch earlier.

The following situations can arise as a result of the difference in switch points (both policies will always select identical starting regions, i.e., the FAR for this model is identical to the Koopman FAR).

(a) $T < T^{**} \leq T^*$, no error, both policies allocate all effort to Region 1.

[1]Under the interpretation of searching for a target with detection capabilities, this result implies that, for identical detectors for both the target and searcher, not necessarily equal, the Koopman policy is optimal. Hence if the decision maker (searcher) is willing to make this assumption, he may proceed as though he were searching for a continuously visible target. This also implies that for an identical visibility process, the Koopman policy is optimal only if $p_1 = p_2$ and $k_2 = k_1$. Recall that in the binary process, the Koopman policy was optimal whenever $v_1 = v_2$.

(b) $T^{**} < T < T^*$, the SKA concentrates all effort in Region 1, while the single interval optimal solution follows (12).

(c) $T^{**} < T^* < T$, both policies split the available effort. The error term in this situation also tends to zero as the available search effort increases without limit, as in Chapter 2.

Figure 3 illustrates some of the above situations. The ordinate again represents the percent relative error, while the abscissa gives the total available search effort. The results of a parametric analysis on the effects of the relative values of the prior probabili s on the error function are given. Note that as the ratio p_1/p_2 tends toward unity the error function decreases.[1] Also note that in each case the peak error occurs at $T = T^*$, the Koopman switch point. The peaks will be analyzed later in this section. It is also interesting to observe the reduction in the error as $\lambda_1/\lambda_2 \to 1$, as was noted earlier. Thus, a decision maker (searcher), operating in this scenario with an amount of search effort T^*, will be required to:

(a) obtain estimates of λ_1 and λ_2 in order to conduct an effective search, or

[1] This suggests that the effect of nonhomogeneous visibility may be lessened in the situation in which the searcher has no prior knowledge on target location. For example, a high likelihood on target position could be offset by a low visibility likelihood.

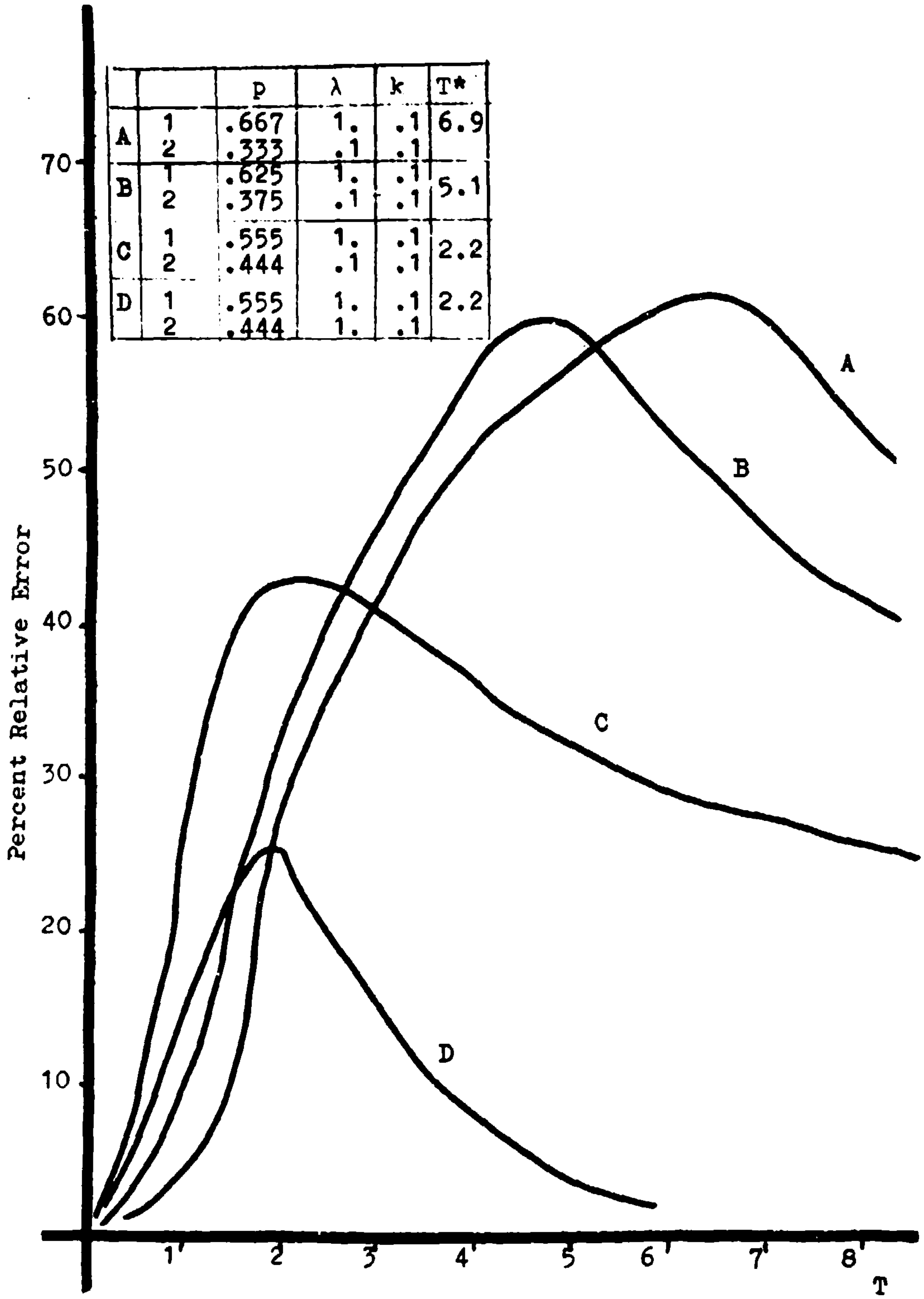


Figure 3 - Error Function Sensitivity

(b) increase the available effort far beyond T*.

An analysis of the error function was undertaken to determine:

(a) At what levels of total available effort the peak differences occur?

(b) What influences their shape, i.e., height, rate of decay, etc.?

For the random-interval, 2-region, initially visible model, one has only two cases to consider, since both policies always select the same initial region and the optimal policy for this model starts splitting the effort before the Koopman policy. These cases are

Case (1) $\frac{\partial E(T)}{\partial T} \leq 0$ for $T > T^*$, and

Case (2) $\frac{\partial E(T)}{\partial T} \geq 0$ for $T > T^*$.

where E(T) denotes the difference in the probability of detection under the two policies. In the first case, we observe that the peak always occurs at T*, since the error will increase until the Koopman policy begins to allocate in Region 2 (recall $T^{**} < T^*$). While in the second case, one must utilize numerical techniques to solve for the time at which the peak difference occurs.[1] The following are

[1]The unimodality of the error function is established by noting that it can be expressed as the difference of two exponential functions.

sufficient conditions for each of the above cases:

Case (1) $$\frac{\lambda_1}{k_1} = \frac{\lambda_2}{k_2}$$

Case (2) $p_1 k_1 = p_2 k_2 \rightarrow T^* = T^{**} = 0.$

This implies that the relationship between the detection and visibility processes is homogeneous across regions. Physically, one might view this as representative of the situation in which whenever the searcher's detection gear is affected by regional conditions, the target's gear is also affected in the same degree. The error function evaluated at the peak for Case (1) is

$$E_p = \frac{p_1 k_1}{\hat{k}_1} \left[\left(\frac{p_1 k_1}{p_2 k_2}\right)^{-\hat{k}_1/k_1} - \left(\frac{p_1 k_1}{p_2 k_2}\right)^{-\frac{\hat{k}_1}{\hat{k}_1+\hat{k}_2}(1+(\hat{k}_2/k_1))} \right]$$

$$+ \frac{p_2 k_2}{\hat{k}_2} \left[1 - \left(\frac{p_1 k_1}{p_2 k_2}\right)^{\frac{\hat{k}_2}{\hat{k}_1+\hat{k}_2}(1-(\hat{k}_1/k_1))} \right] ,$$

where $\hat{k}_i = \lambda_i + k_i$.

Given two search scenarios with the following properties, where the first subscript denotes the region, the second the search situation, $p_{11} = p_{12} = p_{21} = p_{22}$.

$$k_{11} = k_{21},\ k_{12} = k_{22},\ \text{and}\ \frac{\lambda_{11}}{k_{11}} = \frac{\lambda_{12}}{k_{12}},\ \frac{\lambda_{21}}{k_{21}} = \frac{\lambda_{22}}{k_{22}}$$

then the peak error functions for both cases are identical even though the times of occurence are not necessarily identical. Thus as long as the relationship between the detection and visibility process is homogeneous across regions, the decision maker can expect the same maximum error in using the Koopman poliqy, without having knowledge of the visibility parameters.

Since in Case (2) an explicit expression for the error function cannot be given, one must resort to numerical techniques. The following numerical procedure was utilized:

(1) Starting at $T = T^*$ compute $E'(T + \Delta T)$ until that value of T for which $E'(T) \leq 0$.

(2) Use an approximate Fibonacci search on the interval obtained in step (1) to obtain T.

(3) Substitute that value of T into the expression for $E(T)$.

(4) For large values of T (i.e., $T >> T^{**}$), $E(T)$, in this case, is approximated by

$$E(T) = \frac{k_1 p_1}{\hat{k}_1}\left(e^{-\frac{\hat{k}_1 k_2}{k_1 + k_2} T} - e^{-\frac{\hat{k}_1 \hat{k}_2}{\hat{k}_1 + \hat{k}_2} T}\right) + \frac{k_2 p_2}{\hat{k}_2}\left(e^{-\frac{\hat{k}_2 k_1}{k_1 + k_2} T} - e^{-\frac{\hat{k}_1 \hat{k}_2}{\hat{k}_1 + \hat{k}_2} T}\right) \tag{14}$$

Table 3 gives the results for a series of computer runs which were made in an effort to study the sensitivity of the above functions to changes in the search parameters. It contains the measures introduced in Section 2.3.

First we note that in a great many cases the peak difference in the probability of detection is extremely small, suggesting the adequacy of the Koopman policy. The notable exceptions to this observation occur when the visibility parameters are nonhomogeneous. Note also that the assumption $p_1 = p_2$ is not critical relative to the optimality of the Koopman policy in those situations in which $\lambda_1 = \lambda_2$ and $k_1 = k_2$.

The results of Table 3 deal only with the peak errors. One must utilize equation 14 in order to determine their rate of decay with increasing search effort. For example, if $\lambda_1 = 10$, $\lambda_2 = .1$, and $k_1 = k_2 = .1$ (see Table 3), we determine that value of total search effort for which E(T), given by equation 14, is less than .01. Equating E(T) to .01 and solving for T, one obtains $T \approx 28$, and $T^* \approx 7$.

3.2.3 Model Sensitivity

In the following paragraphs, we present the results of a sensitivity analysis on the visibility parameters for both the optimal policy and its associated return. In order

Table 3

Random Interval Sensitivity Results

$p_1/p_2=2$		$T^*=6.9$	$T^*=3.$	$T^*=1.6$	$T^*=0.7$
λ_1	λ_2	$k_1=.1$ $k_2=.1$	$k_1=1.$ $k_2=.1$	$k_1=.1$ $k_2=1.$	$k_1=1.$ $k_2=1.$
.1	.1	.05 1.03 .014	.002 1. .001	.002 1.02 .0006	.001 1. .0004
.1	1.	.016 1.12 .004	.001 1.2 .0006	.13 1.1 .02	.013 1.1 * .01 3.58
.1	10.	.002 1.23 .0005	.025 1.4 * .015 5.72	.72 1.3 .08	.095 1.3 * .05 2.25
1.	.1	.62 2.2 .10	.08 1.1 .028	.001 1.7 .0004	.05 1.1 * .025 1.89
1.	1.	.32 2. .03	.04 1.1 .015	.08 1.11 .014	.05 1. .014
1.	10.	.05 1.2 .003	.007 1. .002	.60 2.1 .05	.016 1.1 .004
10.	.1	.95 3.2 .12	.53 1.6 .07	.12 2.5 * .04 4.3	.60 2.2 .13 .83
10.	1.	.82 7.3 .03	.32 2.7 .03	.02 1.3 .003	.62 2.2 .10
10.	10.	.33 13. .003	.05 2.2 .003	.18 2.5 .007	.32 2. .03
		1 3 2	*Case 2		

1. Relative error at the peak difference.
2. Peak difference & time of occurence, if not equal to T^*.
3. Ratio of conditional expected times (Koopman/optimal).

that the following partial derivatives hold, it is necessary to assume that the available search time is larger than T**. The relevant partials are listed below. Consider first the sensitivity of the optimal policy. For the optimal allocation to region 1, one has

$$\frac{\partial t_1}{\partial \lambda_1} = - \frac{t_1}{(\lambda_1 + k_1 + \lambda_2 + k_2)} \leq 0$$

and

$$\frac{\partial t_1}{\partial \lambda_2} = \frac{t_2}{(\lambda_1 + k_1 + \lambda_2 + k_2)} \geq 0 \ .$$

Symmetrically, for Region 2, we have

$$\frac{\partial t_2}{\partial \lambda_2} = - \frac{t_2}{(\lambda_1 + k_1 + \lambda_2 + k_2)} \leq 0$$

and

$$\frac{\partial t_2}{\partial \lambda_1} = \frac{t_1}{(\lambda_1 + k_1 + \lambda_2 + k_2)} \geq 0 \ .$$

First we observe that the higher the detection and/or visibility rate, the less sensitive the optimal policy to changes in the visibility parameters, i.e., the policy is fairly robust for good detectors and visibility situations in which

the rate is high. It is also apparent, as was *not* the case in the Binary model, that the sensitivity of the optimal policy increases linearly with T, the total available search effort (since t_1 is a linear function of T). Increasing the visibility rate decreases the allocation for that region. Intuitively, this is reasonable since an increase in λ implies a decrease in the mean length of the visible period.

Next we examine the sensitivity of the optimal return function (the probability of detection) to changes in the visibility parameters. In order that the results remain valid, the available search time must be larger than T**. The partial derivatives are listed below. For region 1, we have

$$\frac{\partial P(\tau)}{\partial \lambda_1} = \frac{p_1 k_1}{k_1 + \lambda_1} \left\{ e^{-\alpha\tau}\left[t_1 + \frac{1}{k_1 + \lambda_1}\right]\left(\frac{p_1 k_1}{p_2 k_2}\right)^{-\alpha} - \frac{1}{\lambda_1 + k_1} \right\}$$

and

$$\lim_{\tau\to\infty} \frac{\partial P(\tau)}{\partial \lambda_1} = -\frac{p_1 k_1}{k_1 + \lambda_1}\left(\frac{1}{k_1 + \lambda_1}\right),$$

where

$$\alpha = \frac{\lambda_1 + k_1}{\lambda_1 + k_1 + \lambda_2 + k_2}$$

and

$$\tau = (\lambda_2 + k_2)T.$$

From the symmetry of the two-region search situation, one can obtain $\frac{\partial P(\tau)}{\partial\lambda_2}$ from the above expression by interchanging the regional labels. At T = T**, the above expression becomes

$$\frac{\partial P(T^{**})}{\partial\lambda_1} = \frac{p_1k_1}{k_1 + \lambda_1}\left[\left(\frac{p_2k_2}{p_1k_1}\right)^{-\alpha}\left(T^{**} + \frac{1}{\lambda_1 + k_1}\right)e^{-\alpha T^{**}} - \frac{1}{\lambda_1 + k_1}\right],$$

which is nonpositive if $T^{**} \geq 0$. Thus we observe that the decision maker's analysis of the effects of errors in the visibility parameters rests heavily on the amount of search effort available to him. As $T \to \infty$, the magnitude of $\frac{\partial P(\tau)}{\partial\lambda}$ is the product of two factors:

$\frac{p_1k_1}{k_1 + \lambda_1}$ = the ultimate probability of detection in the i^{th} region,

$\frac{1}{k_1 + \lambda_1}$ = the conditional mean time until detection in the i^{th} region for the single-interval process, conditioned on detection occurring.

Although the optimal policy becomes increasingly sensitive to changes in the visibility parameters with increasing search time, the sensitivity of the return function is limited by the above expressions. Hence, beyond a certain level of total available search time, no matter how great the change in the λ_i, the change in the optimal return is constant.

Furthermore, for large detection and/or visibility rates that constant approaches zero; hence, under these conditions the return is fairly robust. Finally, we use some numerical examples to illustrate the above results. Assume search scenarios characterized by:

#1	#2
$\mathbb{P} = (2/3, 1/3)$	$\mathbb{P} = (2/3, 1/3)$
$\mathbb{K} = (1, 0.1)$	$\mathbb{K} = (10, 1)$
$\lambda = (1,1)$	$\lambda = (10,10)$
$T = 10.$	$T = 2.0$

Then one obtains the following sensitivity results.

#1	#2
$T^{**} = 1.5$	$T^{**} = 0.15$
$t_1 = 4.5$	$t_1 = 0.72$
$t_2 = 5.5$	$t_2 = 1.28$
$KET = 5.7$	$KET = 0.45$
$\frac{\partial t_1}{\partial \lambda_1} = -1.45$	$\frac{\partial t_1}{\partial \lambda_1} = -0.024$
$\frac{\partial t_1}{\partial \lambda_2} = 1.76$	$\frac{\partial t_1}{\partial \lambda_2} = 0.0415$
$\frac{\partial P(T)}{\partial \lambda_1} = -0.167$	$\frac{\partial P(T)}{\partial \lambda_1} = 0.0167$

Summarizing our analysis of the random interval process, we note:

(a) the important differences induced by this visibility process occur under nonhomogeneous visibility conditions (as did those for the Binary model);

(b) while the peak error in using the Koopman policy occurs in limited search time situations, the long-term allocation rates may be sufficiently different from the Koopman rates to induce very slow rates of decay (recall that for the Binary model the long-term allocation rates are identical to those of the Koopman model).

At this point is is of interest to recall that from the results of Stollmack (1968) on visual detection processes, differing detection rates were the rule rather than the exception. Furthermore, the data also indicate heterogeneous visibility conditions over the local terrain.

3.3 Minimization of the Expected Time Until Detection

3.3.1 Model Solution

Recall the discussion of the problem of minimizing the expected time until detection in Chapter 2. The same situation holds for this model, namely, that there is a nonzero probability of failing to detect the target given unlimited search time. The optimal value of the probability of detection for the single interval model, under a constraint on the total search time, is given by (13).

In the limit, equation 6 becomes

$$\lim_{T\to\infty} P(T) = \frac{p_1k_1}{k_1 + \lambda_1} + \frac{p_2k_2}{k_2 + \lambda_2} < 1,$$

for λ_1, $\lambda_2 > 0$.

As in Chapter 2, we consider the conditional expected time to detect given that detection occurs, i.e.,

$$E = \int_0^\infty t \, \frac{dP(t)}{P(\infty)} \quad .$$

Using the procedure of Appendix A, the optimal value of the objective function is

$$E = \frac{1}{P(\infty)} \left\{ \frac{p_1k_1}{k_1 + \lambda_1} \left[\frac{1}{\hat{k}_1}(1 - e^{-\hat{k}_1 T^{**}}) + \left(\frac{p_1k_1}{p_2k_2}\right)^{-\frac{\hat{k}_1}{\hat{k}_1+\hat{k}_2}} \left\{ \frac{\hat{k}_1 + \hat{k}_2}{\hat{k}_1\hat{k}_2} \right\} \left\{ \frac{1 + \hat{k}_1}{\hat{k}_2} \right\} e^{-\frac{\hat{k}_1\hat{k}_2}{\hat{k}_1+\hat{k}_2} T^{**}} \right] + \left[\frac{p_2k_2}{k_2 + \lambda_2} \right] T^{**} \right\} ,$$

where $\hat{k}_i = k_i + \lambda_i$.

For the case in which $p_1k_1 = p_2k_2$, one has that $T^{**} = 0$ and that

$$E = \frac{\hat{k}_1 + \hat{k}_2}{\hat{k}_1\hat{k}_2} = \frac{k_1 + \lambda_1 + k_2 + \lambda_2}{(\lambda_1 + k_1)(\lambda_2 + k_2)} ,$$

which in the limit, as λ_1 & $\lambda_2 \to 0$, tends toward the corresponding result for the Koopman model. This is the conditional

expected time to detect, with probability $(1 - P(\infty))$ the target is not detected.

3.3.2 *Comparison with the Koopman Model*

As in Chapter 2, we consider the case of the partially informed searcher who wishes to minimize the conditional expected time until detection, given unlimited effort and conditioned on detection occurring. The partially informed searcher allocates search effort according to the Koopman policy in an effort to minimize the expected time until detection. The return from such a policy for the single interval model in a 2-region situation is

$$
\begin{aligned}
E\cdot\left(\frac{p_1k_1}{k_1+\lambda_1}+\frac{p_2k_2}{k_2+\lambda_2}\right) &= \frac{p_1k_1}{k_1+\lambda_1}\left\{\frac{\left(1-e^{-\hat{k}_1T^*}\right)}{\hat{k}_1} - T^*e^{-k_1T^*}\right\} \\
&\quad + p_1\left(\frac{p_1k_1}{p_2k_2}\right)^{-\frac{\hat{k}_1}{k_1+k_2}}\left[\frac{k_1}{\hat{k}_1}\right]e^{-\frac{\hat{k}_1k_2T^*}{k_1+k_2}} \\
&\quad \left\{T^* + \frac{k_1+k_2}{\hat{k}_2k_1}\right\} + p_2\left(\frac{p_2k_2}{p_1k_1}\right)^{-\frac{\hat{k}_1}{k_1+k_2}} \\
&\quad \left[\frac{k_2}{\hat{k}_2}\right]e^{-\frac{k_2}{\hat{k}_1+k_2}T^*}\left\{T^* + \frac{k_1+k_2}{\hat{k}_2k_1}\right\},
\end{aligned}
$$

where $T^* = \frac{1}{k_1}\ln\left(\frac{p_1k_1}{p_2k_2}\right)$.

Tables 4-6 contain the results of several comparisons with the Koopman search model, suggesting the following observations:

(a) From Table 3, Section 3.2.3, note that peak relative errors of less than .05 do not, in general, imply that the Koopman policy is "good" for the expected time problem.

(b) For homogeneous visibility rates; the Koopman policy works well for small visibility rates, but poorly for the larger visibility rates (large relative to the detection rates).

(c) For homogeneous detection rates, the Koopman policy works well whenever the visibility rates are either homogeneous or heterogeneous and less than or equal to the detection rates.

(d) At $p_1/p_2 = 10$, note that Koopman policy works well for $\lambda_1 = 0.1$ over all levels of detection rates. Note also the trend, since it begins improving at $p_1/p_2 = 2$. On the other hand, for $\lambda_1 = 10$, in general, the opposite effect occurs implying increasing errors with a increasing prior probability ratio.

Some implications of these results for the decision maker are listed below.

(a) Given a great deal of certainty on target location and a detection rate in the primary region at least as large as the target's visibility rate, the searcher need not be concerned with obtaining estimates of the visibility rates.

Table 4

Random Interval Sensitivity Results: Ratio of Conditional Expected Times Until Detection (Koopman/Optimal)

p1/p2 = 1

λ_1	λ_2	k_1 = .1, k_2 = .1	k_1 = 1., k_2 = .1	k_1 = .1, k_2 = 1.	k_1 = 1., k_2 = 1.
.1	.1	1.	$1.^{+}$	$1.^{+}$	1.
	1.	1.5	1.4	1.1	1.1
	10.	1.9	1.7	1.5	1.7
1.	.1	1.5	1.1	1.4	1.1
	1.	1.	1.1	1.1	1.
	10.	1.6	1.1	2.4	1.5
10.	.1	1.9	1.5	1.7	1.7
	1.	1.6	2.4	1.1	1.5
	10.	1.	2.5	2.5	1.

Table 5

Random Interval Sensitivity Results:
Ratio of Conditional Expected Times
Until Detection (Koopman/Optimal)

pl/p2 = 2

		.1	1.	.1	1.	k_1
λ_1	λ_2	.1	.1	1.	1.	k_2
.1	.1	1.0$^+$	1.0$^+$	1.0$^+$	1.0$^+$	
	1.	1.1	1.2	1.1	1.1	
	10.	1.2	1.4	1.3	1.3	
1.	.1	2.2	1.1	1.7	1.1	
	1.	2.0	1.1	1.1	1.0$^+$	
	10.	1.2	1.0$^+$	2.1	1.1	
10.	.1	3.2	1.6	2.5	2.2	
	1.	7.3	2.7	1.3	2.2	
	10.	13.	2.2	2.5	2.0	

Table 6

Random Interval Sensitivity Results:
Ratio of the Conditional Expected
Times Until Detection (Koopman/Optimal)

pl/p2 = 10

λ_1	λ_2 \ k_1, k_2	.1, .1	1., .1	.1, 1.	1., 1.
.1	.1	1.1+	1.+	1.+	1.+
	1.	1.+	1.+	1.	1.+
	10.	1.+	1.1	1.1	1.1+
1.	.1	3.3	1.1	3.+	1.2
	1.	2.4	1.+	1.8	1.1
	10.	1.2	1.+	1.	1.+
10.	.1	6.+	1.8	9.	3.
	1.	17.3	2.7	7.7	3.3
	10.	16.	1.4	2.8	2.4

(b) On the other hand, if the target's detection rate is greater than that of the searcher, it is important the searcher obtain estimates of it.

One can also make the following observations concerning the implications of these results to the target.

(a) Under complete uncertainty on the searcher's part, the target should choose his visibility rates, assuming they are in some sense under his control, as follows:
   (1) for heterogeneous detection rates, choose homogeneous visibility rates, as large as possible.
   (2) for homogeneous detection rates, choose heterogeneous visibility rates.

(b) If the searcher has increasing prior knowledge on the target's position, the target should use his highest visibility rate in the prime region.

(c) If the maximum available visibility rate in the prime region is less than or equal to the searcher's detection rate in that region, the tactic of remaining in the visible state a random length of time is not effective, at least in terms of ratio of the conditional expected times until detection. Certainly, from a unconditional standpoint, the tactic always has merit inasmuch as it results in a nonzero probability of failing to detect the target.

Chapter 4

A RANDOM INTERVAL OF VISIBILITY: RANDOM INITIATION AND LIMIT

In this chapter we shall consider the situation in which the target is masked when the searcher enters the appropriate region; however, the target becomes visible after some random length of time. The distributions of the starting time and the length of the visible period may be regionally dependent. One may view this as a model of a situation in which the searcher is not sure that the target has arrived in the search zone at the time the search is initiated. The length of the visible period could then correspond to the length of time required for the target to become aware of the searcher's presence.[1]

4.1 Description

Assume that the searcher spends a total of T time units searching a region. Given that the target becomes visible at some time $\mu \varepsilon (0,T)$, the probability of detecting the target is (using the notation of the previous chapter)

[1]The model also has an analogy in a police patrol situation in which the time until a crime is committed is taken to be a random variable as well as the length of time required to carry out the act.

$$\int_{\mu}^{T} \rho(\tau)\bar{H}(\tau)d\tau = \int_{0}^{T-\mu} \rho(s - \mu)\bar{H}(s - \mu)ds$$

Since the above description hinges on the target becoming visible at μ (which is itself a random variable with probability density function $f(\mu)$), the conditional probability of detection, conditioned on the target presence, is given by

$$P(T) = \int_{0}^{T} f(\mu) \int_{0}^{T-\mu} \rho(s - \mu)\bar{H}(s - \mu)ds d\mu \qquad (15)$$

For an N-region search situation, the problem of maximizing the probability of detecting the target, given T time units, is

$$\max \sum_{i=1}^{N} p_i \int_{0}^{t_i} f(\mu) \int_{0}^{t_i-\mu} \rho(s - \mu)\bar{H}(s - \mu)ds \; d\mu \qquad (16)$$

$$\text{S.T.} \sum_{i=1}^{N} t_i \leq T, \quad t_i \geq 0 .$$

The following special cases are of interest: taking $f(\mu)$ as an impulse function at $\mu = 0$ yields the model of Chapter 3; setting $\bar{H}(\tau) = 1$ for $\tau \geq 0$, yields the situation in which the target arrives late, but never departs (the model of Blackman (1959)). In this chapter h(t) has an infinite domain, situations in which $h(\tau)$ or both $f(\tau)$ and $h(\tau)$ have finite domains are considered in Appendix D.

The solutions for this model depend upon the distributions of the time until detection, visibility period, and the start of the visibility period. We have already remarked on the appropriateness of

(a) exponential detectors

(b) an exponential distribution on the length of the visible period.

We shall consider two examples of start-time distributions.

Consider first the uniform distribution of starting times in the i^{th} region, given by

$$f_i(\mu) = \begin{cases} \frac{1}{S_i}, & 0 \leq \mu \leq S_i \\ 0, & \mu > S_i \end{cases} .$$

Referring to equation 15, and dividing the region of integration into the appropriate parts one obtains[1]

[1] Formulations for discrete detectors are given in Appendix B. Continuous search space versions of the model are discussed in Appendix C.

$$P_i(t) = \begin{cases} \frac{1}{S_i}\left(\frac{k_i p_i}{\lambda_i + k_i}\right)\left[t - \frac{(1 - e^{-(\lambda_i+k_i)t})}{(\lambda_i + k_i)}\right], \\ \qquad t \le S_i , \\ \frac{1}{S_i}\left(\frac{k_i p_i}{\lambda_i + k_i}\right)\left[S_i - \frac{(e^{-(\lambda_i+k_i)(t-S_i)} - e^{-(\lambda+k)t})}{\lambda_i + k_i}\right] \end{cases} \quad (17)$$

for $t > S_i$

Note that the limit of (17) as $t \to \infty$ is

$$\lim_{t\to\infty} P_i(t) = \frac{k_i p_i}{\lambda_i + k_i} ,$$

which is equal to the limiting result for the model of Chapter 3. Finally we note that since

$$\frac{\partial^2 P(t)}{\partial t^2} = \begin{cases} \frac{k_i p_i}{S_i} e^{-(\lambda_i + k_i)t} \geq 0 \ , \quad t \leq S_i \ , \\ -\frac{k_i p_i}{S_i}\left[e^{-(\lambda_i+k_i)t}(e^{(\lambda_i+k_i)S_i} - 1)\right] \ , \quad t > S_i \ , \end{cases}$$

the conditional detection function is convex until $t = S_i$ and concave thereafter. We refer to this type of function as *pseudo-concave*.[1] Consider next the situation in which the start of the visibility period is exponentially distributed, i.e., $f(\mu) = \beta e^{-\beta\mu}$ for $\mu \geq 0$. Employing equation 15 for the i^{th} region one obtains

$$P(t) = \begin{cases} \frac{k_i p_i}{\lambda_i + k_i}\left[1 - \frac{(\beta_i e^{-(\lambda_i + k)t} - (\lambda_i + k_i)e^{-\beta_i t})}{(\beta_i - (\lambda_i + k_i))}\right] , \\ \qquad \text{assuming } \beta_i \neq \lambda_i + k_i , \\ \frac{k_i p_i}{\lambda_i + k_i}\left[1 - e^{\beta_i t}(1 + \beta_i t)\right] \ , \\ \qquad \text{if } \beta_i = \lambda_i + k_i \ . \end{cases} \tag{18}$$

[1] These have been explored by Zahl (1963) and Onaga (1971); however, no attempt was made to provide examples of search situations which would yield such functions.

One can show that (assuming $\beta_i \neq \lambda_i + k_i$)

$$\frac{\partial^2 P(t)}{\partial t^2} \geq 0 \text{ for } t \leq \bar{T} = \frac{1}{\beta_i - (\lambda_i + k_i)} \ln\left(\frac{\beta_i}{\lambda_i + k_i}\right),$$

and is negative for $t > \bar{T}$.

while for $\beta_i = \lambda_i + k_i$, the point $\bar{T}$ is given by

$\bar{T} = \frac{1}{\beta_i}$ = mean length of time until the target becomes visible.

4.2 *Allocation of Search Effort to Maximize the Probability of Detection*

4.2.1 *Model Solution*

Consider first the situation in which the start time is uniformly distributed on [0,S] (see equation 17). An exact analytic (explicit) solution for this model cannot be attained.[1] The structure of the discrete search space version of the problem of maximizing the probability of detection under a constraint on the available search time is ideally suited for a

[1] This is primarily due to the fact that for $T < S$, $P(T)$ does not possess an inverse function.

dynamic programming approach. Hence, any subsequent numerical results for this model (regardless of the assumption on start times) will have been determined via dynamic programming, unless otherwise noted.[1]

Several important features can be determined from a marginal analysis of the model. The First Allocation Rule for this case may be stated as: Choose the region j for which

$$\frac{p_j k_j}{S_j} = \max_{1 \le i \le N} \frac{p_i k_i}{S_i} .$$

While we are unable to specify an analytic solution in the more general situation, it is possible to determine the solution provided T is large enough so that $t_1 > S_1$ and $t_2 > S_2$ in the 2-region case. The conditional detection probability can be written as

$$P(t_i) = \frac{k_i}{k_i + \lambda_i} \left\{ 1 - e^{-(\lambda_i + k_i)t_i} \left[\frac{e^{(\lambda_i + k_i)S} - 1}{(\lambda_i + k_i)S} \right] \right\}.$$

Setting

$$d_i = \frac{e^{(\lambda_i + k_i)S_i} - 1}{(\lambda_i + k_i)S_i} ,$$

---

[1] The program descriptions and listings are contained in Appendix E.

which is a constant, we note that the above function differs from that of the previous chapter by the constant d_i (independent of t_i). Accordingly, the techniques discussed in Appendix A are applicable here and obtains for the 2-region case

$$
\begin{aligned}
t_1 &= \frac{1}{\lambda_1 + k_1 + \lambda_2 + k_2} \left\{ \ln \frac{p_1 k_1 d_1}{p_2 k_2 A_2} + (\lambda_2 + k_2) T \right\} , \\
t_2 &= \frac{1}{\lambda_1 + k_1 + \lambda_2 + k_2} \left\{ \ln \frac{p_2 k_2 d_2}{p_1 k_1 A_1} + (\lambda_1 + k_1) T \right\} .^{1}
\end{aligned}
\tag{19}
$$

Extension to the N-region case is also easily obtained via the results of Appendix A.

Consider next the case of the exponential distributions leading to the conditional detection functions given by (18). The First Allocation Rule (FAR) in this case is given as: Choose the region j for which

$$p_j k_j \beta_j = \max_{1 \leq i \leq N} p_i k_i \beta_i$$

We also note that as $\beta_j \to +\infty$ for every j, this model tends toward that of Chapter 3. An interesting property of the

[1]Note the similarity to the optimal allocations for the model of Chapter 3.

pseudo-concave detection function, is that (in the case of identical regions) one doesn't start the allocation procedure by dividing the search effort equally among the identical regions as in the SKA or any other model with a concave conditional detection function (see Figure 8).

Again, while we are unable to determine an analytic solution in the most general situation, it is possible to approximate it provided one can make the following assumptions. First we note that for large quantities of search effort, (18), under the assumption $(\lambda + k) > \beta$, becomes

$$P(T) \approx \frac{kp}{k + \lambda}\left[1 - \frac{(\lambda + k)e^{-\beta T}}{(\lambda + k) - \beta}\right] . \tag{20}$$

Also if one has $(\lambda + k) >> \beta$, then the requirement of large quantities of search effort may be dropped and (20) becomes

$$P(T) \approx \frac{kp}{k + \lambda}\left[1 - e^{-\beta T}\right] . \tag{21}$$

Next we observe that in the situation in which $\beta > \lambda + k$ or $\beta >> (\lambda + k)$ the analogous equations (from (18)) become

$$P(T) \approx \frac{pk}{k + \lambda}\left[1 - \frac{e^{-(\lambda+k)T}}{\beta - (\lambda + k)}\right], \tag{22}$$

and

$$P(T) \approx \frac{pk}{k + \lambda}\left[1 - e^{-(\lambda+k)T}\right] . \tag{23}$$

In an N-region search problem, there are 2^N possible cases to consider. Here we shall exhibit one possible result for the 2-region situation.

Certainly, under both of the above assumptions, the resulting optimization problem has the same general form as those studied in Appendix A. In a 2-region search problem where $(\lambda_1 + k_1) > \beta_1$ and $\beta_2 > (\lambda_2 + k_2)$, one obtains for the approximate allocation scheme

$$t_1 = \frac{1}{(\lambda_2 + k_2 + \beta_1)}\left[\ln [z_1/z_2] + (\lambda_2 + k_2)T\right], \qquad (24)$$

and $t_2 = T - t_1$. Assuming $z_1 > z_2$, the approximate switch points are given by

$$T^* = \frac{1}{\beta_1} \ln [z_1/z_2] ,$$

where

$$z_1 = \frac{k_1 p_1 \beta_1}{(k_1 + \lambda_1) - \beta_1} \quad \text{and} \quad z_2 = \frac{k_2 p_2 \beta_2}{\beta_2 - (\lambda_2 + k_2)} .$$

The results for the other three cases in the 2-region problem follow from the appropriate substitutions into (24). Likewise, the extension to the N-region case is obtained via the results of Appendix A.

Assuming that

$$\lim_{T\to\infty} \frac{t_1}{t_2} \to \phi, \text{ a constant,}$$

for the 2-region case in which $\beta_i = \lambda_i + k_i$, one obtains as approximate allocation policies

$$t_1 = \frac{1}{\beta_1 + \beta_2}\left[\ln \frac{p_1 k_1}{p_2 k_2} + 2 \ln \frac{\beta_1}{\beta_2} + \beta_2 T\right]$$

and

$$t_2 = \frac{1}{\beta_1 + \beta_2}\left[\ln \frac{p_2 k_2}{p_1 k_1} + 2 \ln \frac{\beta_2}{\beta_1} + \beta_1 T\right] .$$

Finally, we note that the above results can be specialized to the situation in which the target arrives at some random time after the start of the search and doesn't leave or become masked. One need only choose $\lambda_i = 0$ in each of the above examples, and, in general, assume that

$$\bar{H}(\tau) = 1, \text{ for } \tau \geq 0.$$

4.2.2 Comparison with the Koopman Model

In this section we examine the situation in which a partially informed searcher applies the Koopman policies to situations in which the target behavior is actually characterized by the random interval of visibility process. Recall that the First Allocation Rule for the Koopman search problem is

Choose the region j for which

$$p_j k_j = \max_{1 \le i \le N} p_i k_i ,$$

while for the uniform-start-time model, one has as the FAR,

Choose the region j for which

$$\frac{p_j k_j}{S_j} = \max_{1 \le i \le N} \frac{p_i k_i}{S_i} .$$

Thus one could have the following situations:

(a) $p_1 k_1 > p_2 k_2$ and $S_1 > S_2$, which under SKA implies that Region 1 is selected first, but under the FAR for this model, assuming $\frac{p_1 k_1}{S_1} < \frac{p_2 k_2}{S_2}$, Region 2 is selected first.

(b) $p_1 k_1 > p_2 k_2$, $\frac{p_1 k_1}{S_1} > \frac{p_2 k_2}{S_2}$ and $T^* < S_1$, thus

(1) for $T \varepsilon [0,T^*]$ no error

(2) $T \varepsilon [T^*,S_1]$, SKA divides effort while the single interval policy allocates to Region 1 only.

(3) $T > S_1$, both policies divide the effort.

Figure 4 illustrates the respective policies in the situation in which, for T sufficiently small, a searcher using the Koopman policies will place all the search effort in the

wrong region. It also illustrates a phenomenon heretofore unencountered in the search literature, vis., an optimal policy which is decreasing with increasing total available effort.[1] In the example shown, the optimal policy is such that for small values of search time all the effort is placed in Region 1, and finally for larger levels of search time the optimal policy begins to divide the time. We note that the Koopman policy is nondecreasing with increasing effort, as shown in Figure 4. Figure 5 is a plot of the percent relative error in the probability of detection when the Koopman policy is used for the allocations described in Figure 4. Note that for the levels of visibility parameters of Figure 4, once sufficient search effort is available, the parameters satisfy the conditions noted in Section 3.2.2 for agreement with the Koopman allocation--hence, the rapid decay of the error function.

One can make use of the analytic expression obtained in Section 4.2.1 to exhibit an approximate error function. For 2-regions, one obtains as an approximate error expression,[2]

---

[1] An exception to this statement is the work of Kronz (1971) conducted simultaneously with this effort at the Systems Research Laboratory, The University of Michigan.

[2] Note the similarity between (25) and the error function for the model of Chapter 3, equation 14.

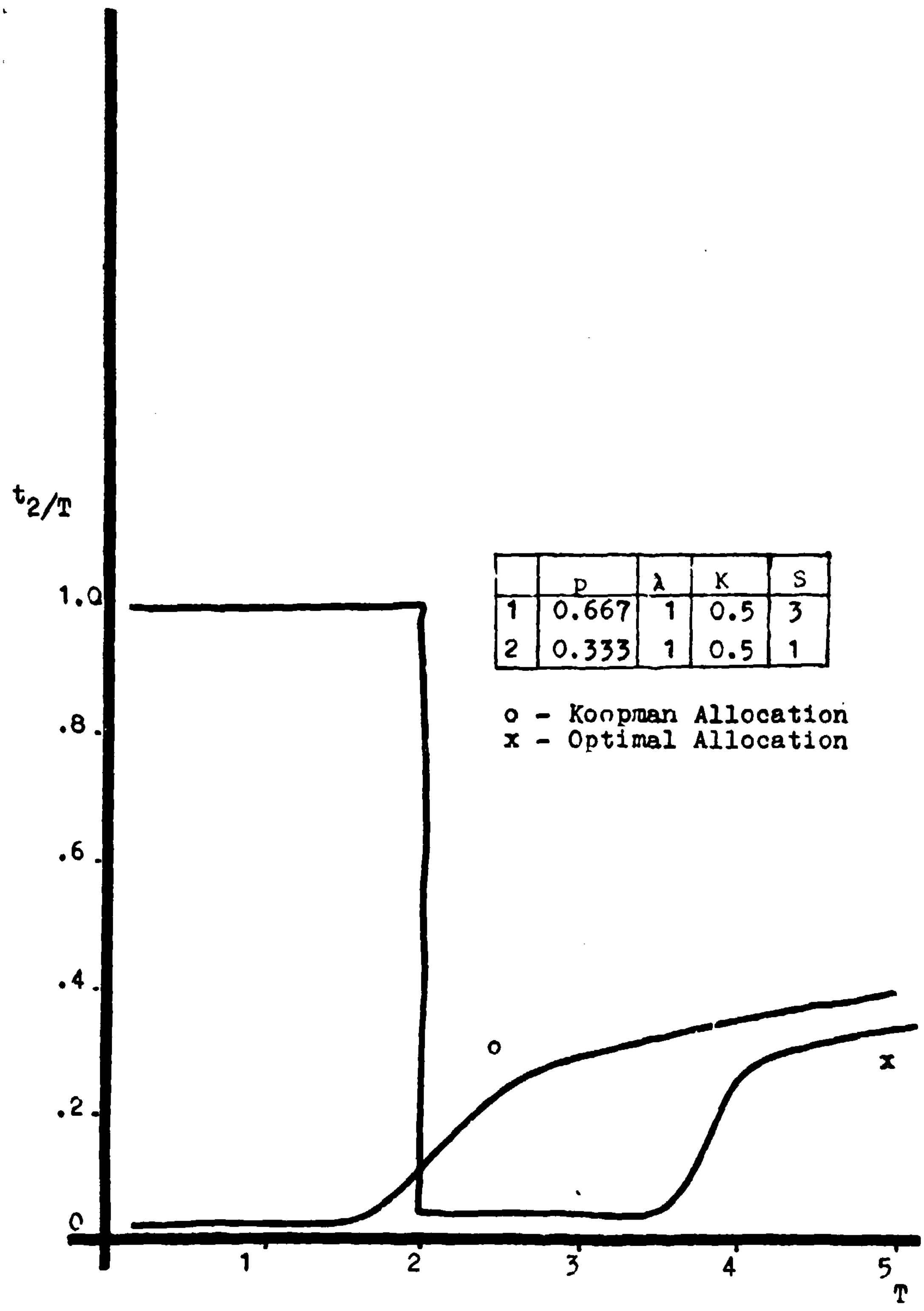
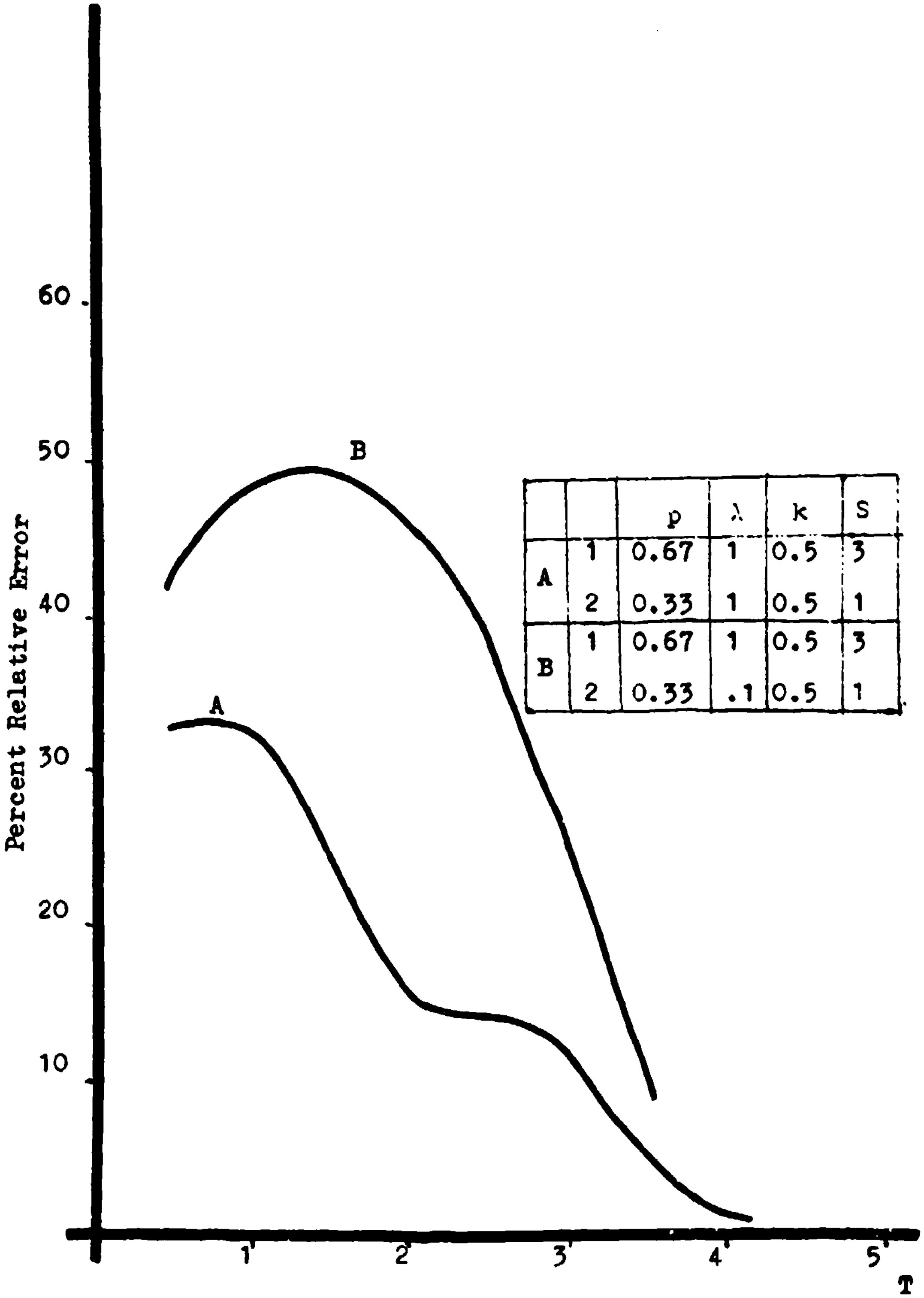


Figure 4 - A Comparison of Allocation Policies



		p	λ	k	S
A	1	0.67	1	0.5	3
	2	0.33	1	0.5	1
B	1	0.67	1	0.5	3
	2	0.33	.1	0.5	1

Figure 5 - Percent Relative Error versus Search Time

$$E(T) \approx \frac{p_1 k_1 d_1}{k_1 + \lambda_1}\left[e^{-\frac{(\lambda_1+k_1)k_2 T}{k_1 + k_2}} - e^{-\frac{(\lambda_1+k_1)(\lambda_2+k_2)T}{\lambda_1+k_1+\lambda_2+k_2}} \right] + \frac{p_2 k_2 d_2}{k_2 + \lambda_2}\left[e^{-\frac{(\lambda_2+k_2)k_1 T}{k_1 + k_2}} - e^{-\frac{(\lambda_1+k_1)(\lambda_2+k_2)T}{\lambda_1+k_1+\lambda_2+k_2}} \right] \quad (25)$$

We use the term "approximate" because in developing the above expression it was assumed that the total available effort, T, was allocated as follows by the respective policies:

Koopman	Single Interval
$t_1 \approx \frac{k_2}{k_1 + k_2} T$	$t_1 \approx \frac{(\lambda_2 + k_2)T}{k_2 + \lambda_2 + k_1 + \lambda_1}$
$t_2 \approx \frac{k_1}{k_1 + k_2} T$	$t_2 \approx \frac{(\lambda_1 + k_1)T}{k_2 + \lambda_2 + k_1 + \lambda_1}$

(26)

These allocations were obtained from equations 19 by neglecting the constant terms.

Considering the exponential-start-time model, recall that the First Allocation Rule is

Choose the region j for which

$$p_j k_j \beta_j = \max_{1 \le i \le N} p_i k_i \beta_i .$$

The combination of differing FAR and differing switch points can lead to the following situations when one searches according to the Koopman policy.

(a) $p_1k_1 > p_2k_2$, $\beta_2 > \beta_1$, and $\beta_1p_1k_1 < \beta_2p_2k_2$, thus the SKA chooses the wrong region under an extremely constrained search time situation.

(b) Or, one could have $\beta_1p_1k_1 > \beta_2p_2k_2$ and $p_1k_1 > p_2k_2$ and

$$\bar{T} >> T^*.$$

In this case, one has

(1) $t \in [0,T^*]$, no error

(2) $t \in [T^*,\bar{T})$, a division of effort under SKA, all effort in region (1) under the optimal policy,

(3) $t \in [\bar{T},\infty)$, a division of effort under both policies.

Figure 6 is a plot of percent relative error in the probability of detection under the SKA policy, and Figure 7 gives the probability of detection under the optimal policy.

Figure 6 presents the results of an analysis on the effects of variations in the rates of the process controlling the length of time until the target becomes visible. Observe that as the respective rates decrease, the percent relative error curve increases everywhere; and that in each case, a searcher using the Koopman policy chooses the wrong initial region. Also note that at $T \simeq T^*$, the Koopman policy begins to allocate some effort to Region 1 and the relative error function begins to decrease. Figures 6 and 7

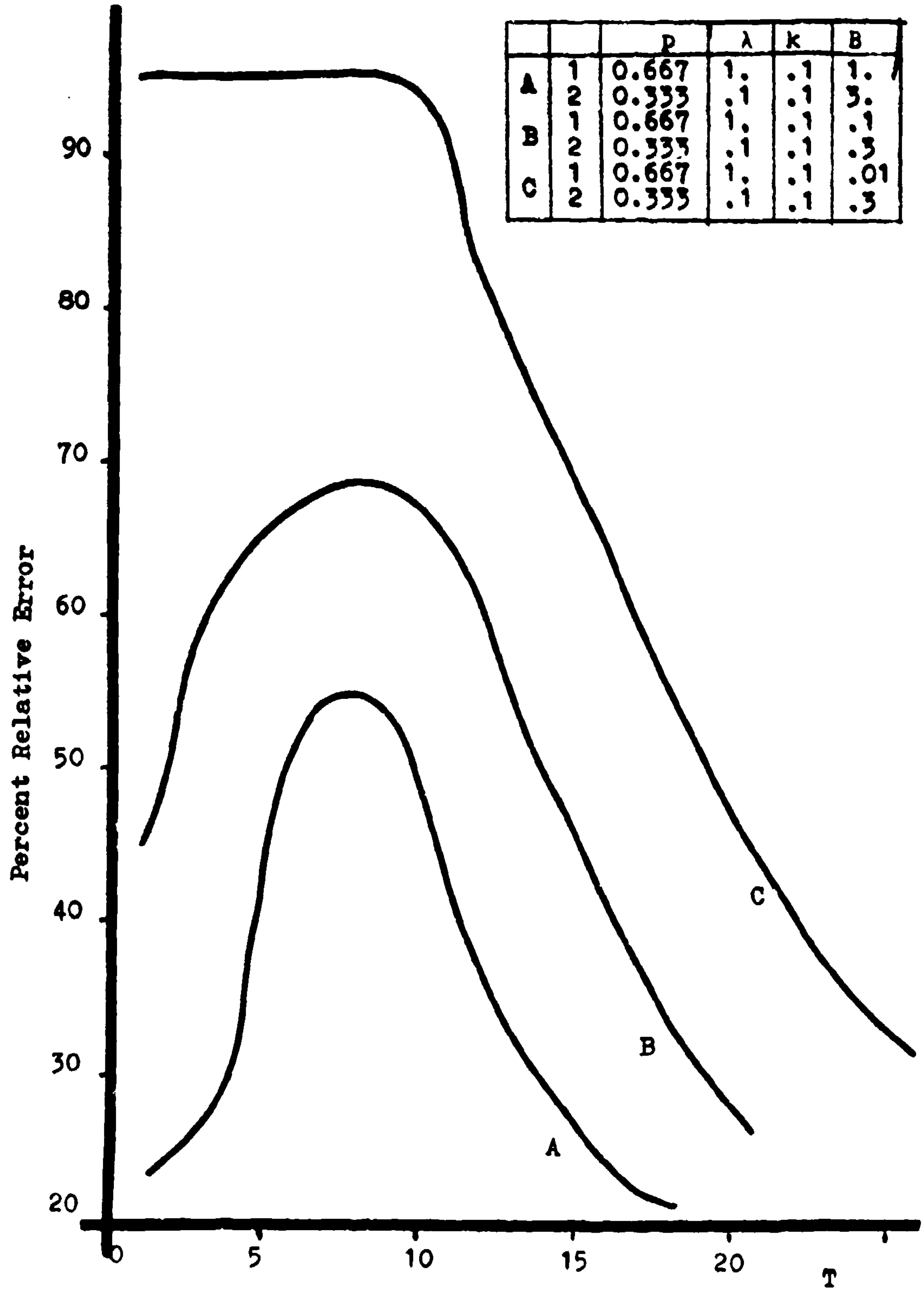


Figure 6 - Error Function Sensitivity

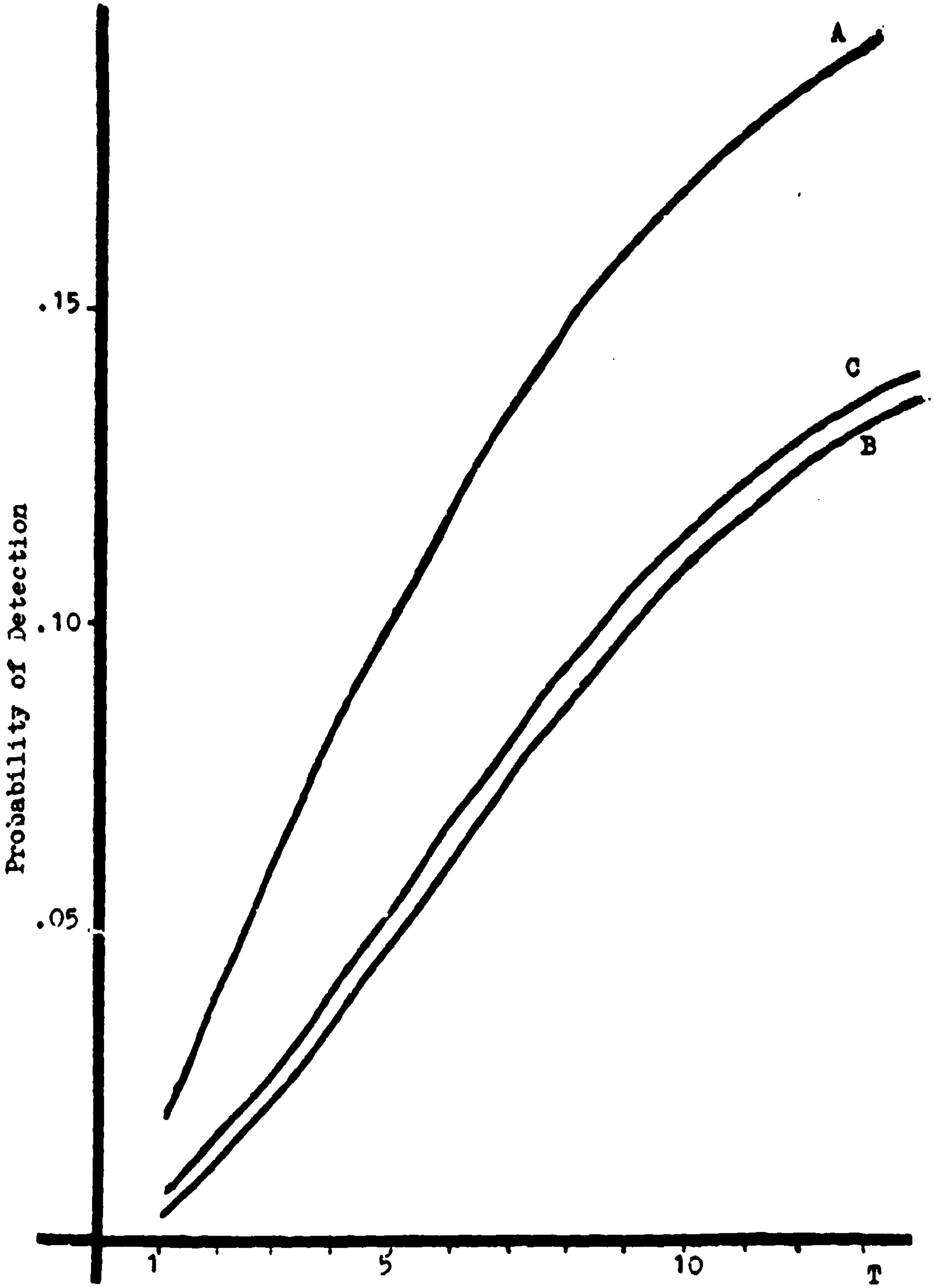


Figure 7 - Optimal Return Versus Search Time

enable one to determine the actual magnitudes of these errors, in every case the limiting value of the probability of detection is 0.2265 (obtained from equation (16) when $T \to \infty$).

Figure 8 contains some results which are characteristic of the pseudo-concave detection functions for the random interval model. The figure depicts the optimal allocation policy for a 4-region problem in which the regional search parameters are identical. The Koopman policy in this situation divides the search time evenly among the regions, but in the random interval model such is not the case. For the 2-region case in which

$$(\lambda_1 + k_1) > \beta_1, \ \beta_2 > (\lambda_2 + k_2) ,$$

and T is large, one obtains the following approximate error function[1]

$$E(T) = \frac{p_1 k_1}{k_1 + \lambda_1}\left\{ e^{-\frac{\beta_1 k_2 T}{k_1 + k_2}} - e^{-\frac{\beta_1(\lambda_2 + k_2)T}{\beta_1 + \lambda_2 + k_2}} \right\} + \frac{p_2 k_2}{k_2 + \lambda_2}\left\{ e^{-\frac{(\lambda_2 + k_2)k_1 T}{k_1 + k_2}} - e^{-\frac{\beta_1(\lambda_2 + k_2)T}{\beta_2 + \lambda_2 + k_2}} \right\} . \tag{27}$$

[1] One can interpret $\lambda_i + k_i$ as the combined detection rate under the visibility process, hence $1/(\lambda_i + k_i)$ is the conditional mean time until detection, conditioned on detection occuring in that region. Thus the approximate allocation policies are always determined by using the smaller of the two rates, the combined detection rate, and the rate for the start of the visible period.

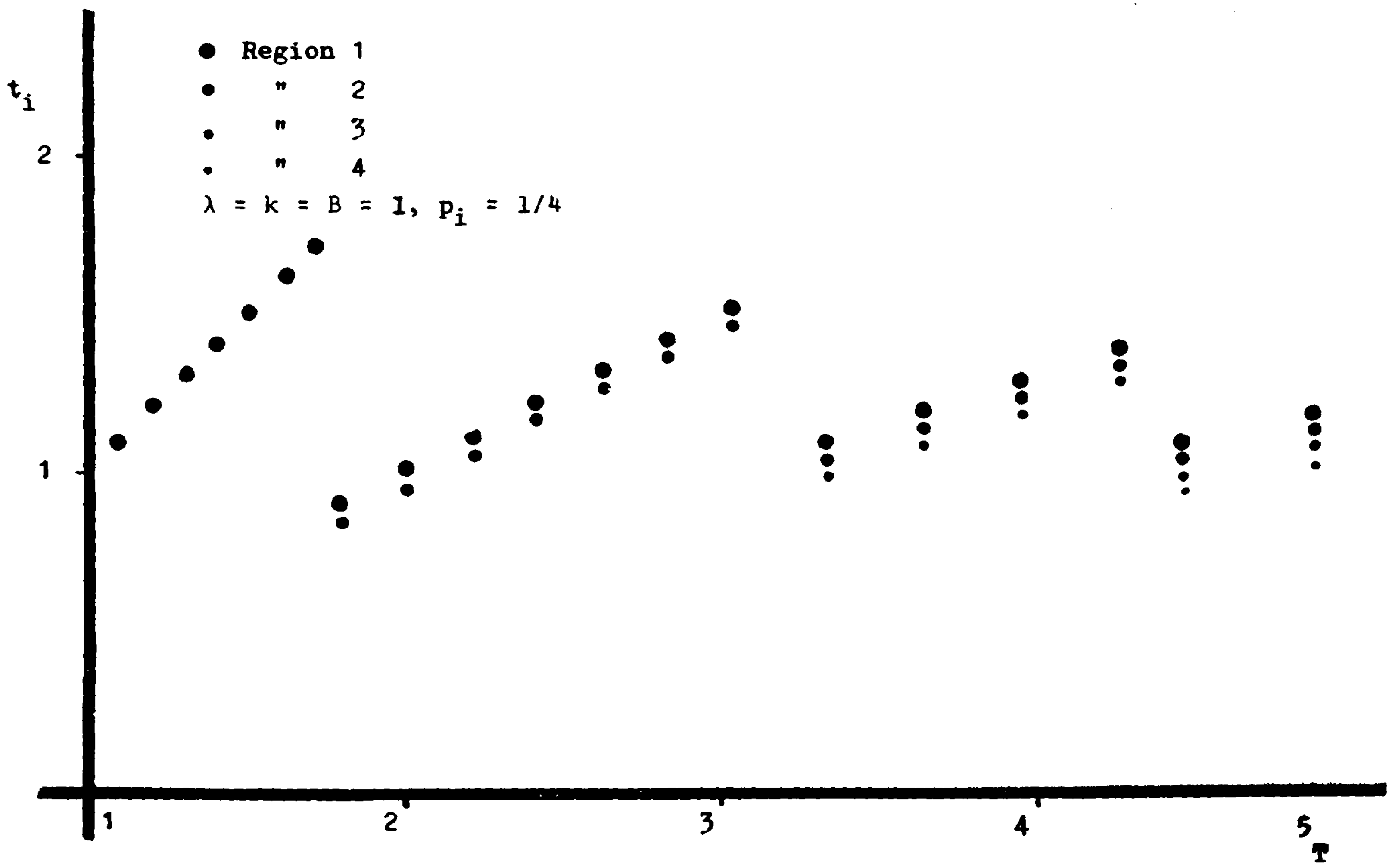


Figure 8 - The Allocation Policy for Four Identical Regions

Again, in deriving the above expression it was assumed that the total search time T was allocated as follows by the respective policies (see equations 24):

Koopman	Single Interval
$t_1 = \dfrac{k_2 T}{k_1 + k_2}$	$t_1 = \dfrac{(\lambda_2 + k_2)T}{\beta_1 + (\lambda_2 + k_2)}$
$t_2 = \dfrac{k_1 T}{k_1 + k_2}$	$t_2 = \dfrac{\beta_1 T}{\beta_1 + (\lambda_2 + k_2)}$

(28)

Equation 27 can be used to study the sensitivity of the error function to changes in the search parameters as well as to determine the amount of search time required to reduce the error function to some specified value.

Figure 9 is an example of the situation in which the proper combination of parameters cam yield an error function which continues to increase over a large range of total available search effort. In Figure 9, one has $\lambda_1 + k_1 > \beta_1$ and $\lambda_2 + k_2 > \beta_2$, hence, as we have already observed, the approximate long-term allocations become

$$t_1 = \frac{\beta_2}{\beta_1 + \beta_2} T \text{ and } t_2 = \frac{\beta_1}{\beta_1 + \beta_2} T \quad ,$$

which for this situation are the reverse of the long-term Koopman allocation policy. In other words the dominant process in this example is the length of time until the target becomes visible.

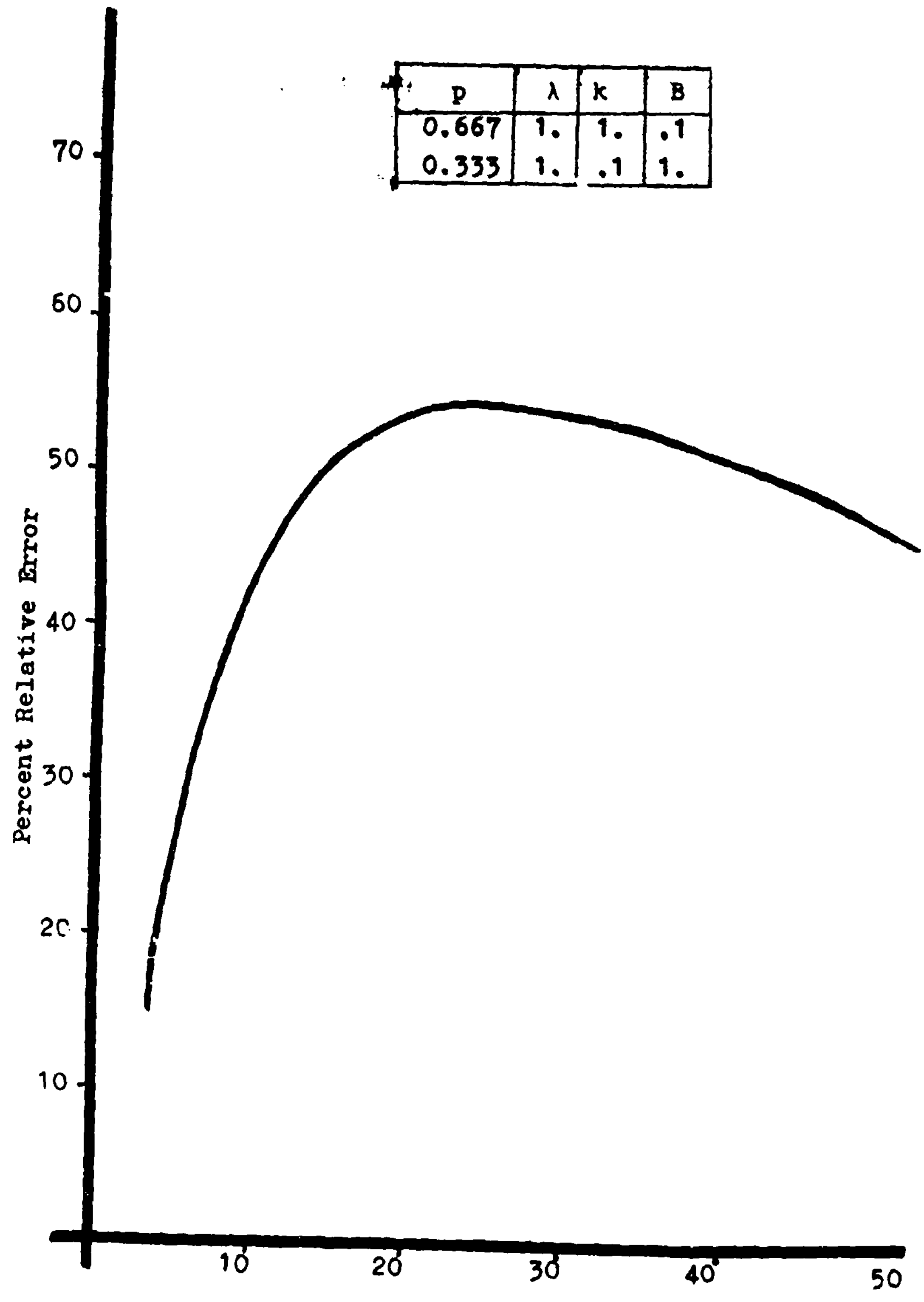


Figure 9 - Percent Relative Error Versus Search Time

4.8.3 Model Sensitivity

In this section we present the results of an analysis of the sensitivity of both the optimal policy and its associated return to changes (or errors) in the visibility parameters. The approximate allocation policies (equation 26) and the resulting returns obtained in Section 4.2.2 are used. Thus we are implicitly assuming that sufficient search time is available for these approximations to hold.

As was noted, the optimal policy for the uniform start time model, under the assumption that T >> 1, is identical to that for the model of Chapter 3. Hence we need only consider the exponential start time model.

For the exponential-start-time model, the sensitivity of the optimal allocations is determined from the approximate solution. For the example of this section (i.e., $(\lambda_1 + k_1) > \beta_1$ and $\beta_2 > (\lambda_2 + k_2)$), one has the following partial derivatives:

$$\frac{\partial t_1}{\partial \beta_1} = -\frac{t_1}{\lambda_2 + k_2 + \beta_1}, \qquad \frac{\partial t_1}{\partial \lambda_2} = \frac{t_2}{(\lambda_2 + k_2 + \beta_1)}$$

$$\frac{\partial t_2}{\partial \lambda_2} = -\frac{t_2}{\lambda_2 + k_2 + \beta_1}, \qquad \frac{\partial t_2}{\partial \beta_1} = \frac{t_1}{(\lambda_2 + k_2 + \beta_1)}.$$

Since the t_i are linear functions of the total available search effort, the sensitivity of the optimal allocations to the visibility parameters increases with increasing available search effort. In view of the assumptions required for the approximate solution, i.e., $\lambda_1 + k_1 > \beta_1$ and $\beta_2 > \lambda_2 + k_2$, for a fixed

level of search time, the optimal policy will be highly sensitive to changes in the relevant visibility parameters λ_2 and β_1.

The ramifications of the sensitivity of the optimal policy must be evaluated in light of the robustness of the return function, since we are only considering situations in which the total search time is large enougn to permit one to utilize the approximate solutions. Accordingly, the partial derivatives of the return function with respect to the visibility parameters are given by

$$\frac{\partial P(T)}{\partial \beta_1} = \frac{(\lambda_2 + k_2)^2 T}{(\beta_1 + \lambda_1 + k_2)^2} e^{-\frac{\beta_1(\lambda_2+k_2)}{\beta_1+\lambda_2 k_2} T} \left\{ \frac{p_1 k_1}{\lambda_1 + k_1} + \frac{p_2 k_2}{\lambda_2 + k_2} \right\}$$

$$\geq 0$$

$$\frac{\partial P(T)}{\partial \lambda_2} = - \frac{p_2 k_2}{(\lambda_2 + k_2)^2} \left(1 - e^{-\frac{\beta_1(\lambda_1+k_1)T}{\lambda_2+k_2+\beta_1}} \right) + \frac{\beta_1 t_2}{\beta_1 + \lambda_2 + k_2}$$

$$\cdot \left\{ \frac{p_1 k_1}{\lambda_1 + k_1} + \frac{p_2 k_2}{\lambda_2 + k_2} \right\} e^{-\frac{\beta_1(\lambda_2+k_2)T}{\beta_1+\lambda_2+k_2}} .$$

As $T \to \infty$, one has

$$\frac{\partial P(T)}{\partial \beta_1} = 0 \text{ and } \frac{\partial P(T)}{\partial \lambda_2} = - \frac{p_2 k_2}{(\lambda_2 + k_2)^2} .$$

Intuitively, the signs of these partials are reasonable since (a)

increasing β_1 decreases the mean time until the target becomes visible in the first region allowing (in a fixed total time T) more time for searching while the target is visible, thus increasing the probability of detection; (b) increasing λ_2 has just the opposite effect in region 2, thus reducing the overall probability of detection. Thus, although the optimal policy becomes increasingly sensitive to changes in the visibility parameters with increasing search time, the sensitivity of the return function is limited by the above expressions. Hence, beyond a certain level of search effort, no matter how great the changes in the optimal policy due to changes in the λ_i, the change in the optimal return is essentially constant. Thus whenever λ and k are large it is robust.

Suppose the searcher had some means of controlling the visiblity parameters with an associated cost, but not eliminating them. The above expressions yield the loss or gains in the return function per unit change in these parameters. On the other hand, the decision maker could weigh this action against that of using additional search time and then determine the most cost-effective option. To illustrate the relative magnitudes of these partial derivatives, we consider the search situation determined by $\mathbb{P} = (2/3, 1/3)$, $\mathbb{K} = (1, .1)$, $\lambda = (1, .1)$, $\beta = (.1, 1)$, one has then

$$\left.\frac{\partial t_1}{\partial \lambda_2}\right|_{T=20} = 22.2 , \quad \left.\frac{\partial t_2}{\partial \beta_1}\right|_{T=20} = 44.4 ,$$

$$\left.\frac{\partial P(T)}{\partial \lambda_2}\right|_{T=20} = -0.122 \text{ and } \left.\frac{\partial P(T)}{\partial \beta_1}\right|_{T=20} = 1.18 .$$

Thus in situations in which the detection and visibility rates are low, both the optimal policy and return are sensitive to changes in the visiblity parameters.

Appendix F contains the description and results of a numerical study of the error function over the entire range of total available search effort in contrast to the above study of the approximate results. These results are briefly summarized as follows. In contrast to the models of Chapters 2 and 3,

(a) the use of the Koopman policy will not be adequate in situations in which the visibility conditions are homogeneous across the regions of interest,

(b) the error function is not reduced by increasing the available search time to some realistic level,

(c) the maximum errors do not occur in situations in which the total available search time is highly constrained (i.e., less than or equal to the Koopman switch time).

(d) the availability of extremely good detectors (high rates) will not imply that the Koopman policy will yield small errors.

These results imply that the decision maker, in general, must obtain accurate estimates of the visibility parameters in order to conduct an effective search, since his option of

increasing the available search time may no longer be cost-effective.

4.3 Minimization of the Expected Time Until Detection

4.3.1 Model Solution

In its most general form this model has the property that the limiting value of the probability of detection under the optimal policy is less than unity. Thus the expected time to detect the target increases without limit. However, one can, as in Section 3.3.1, consider the conditional expected time until detection, conditioned on detection occurring. We consider first the special case in which the target may arrive (or appear) at some random time after the start of the search and remain (stay visible). From the general expressions of the detection probabilities of Section 4.1 we observed that for $T \to \infty$, the limiting value of the probability of detection is unity. For the purpose of the following discussion, we restrict attention to this case.

Recall the discussion of the early results of Dobbie (1963) in Chapter 1. Namely, that one can always express the expected time until detection as

$$E = \int_0^\infty (1 - P(\phi(T))dT,$$

where P(T) = probability of detection under the allocation policy $\phi(T)$. If $\phi(T)$ is nondecreasing in T, and maximizes

$P(\phi(T))$ for each value of T, it also is the optimal policy for the expected time problem. From the examples of Section 4.2.2, we recall that the optimal policies for the maximization of the probability of detection did not necessarily have the property that the amount allocated a given region was non-decreasing as a function of the available search effort (see Figure 4). The reason for this was the fact that the detection functions for this model are pseudo-concave. Thus from the above we may conclude that the optimal allocation policies for the two search objectives are different, at least for small values of search effort.

How then does one generate the optimal policy for the expected time problem in this situation? Intuitively, it is clear that the two policies will have to agree almost everywhere. Onaga (1971) studied the most general form of the expected time problem. As a result of his work, one may obtain the optimal policy by solving the problem of maximizing the probability of detection using a modified form of the detection function. In particular, the form used is the "minimal concave majorant[1]" of the detection function which is constructed as follows. Graphically the majorant function for a pseudo-concave function (dashed curve) is shown as the solid curve in the figure below.

[1]This term was apparently defined by Stone (1971) since it is not widely used in the mathematical literature.

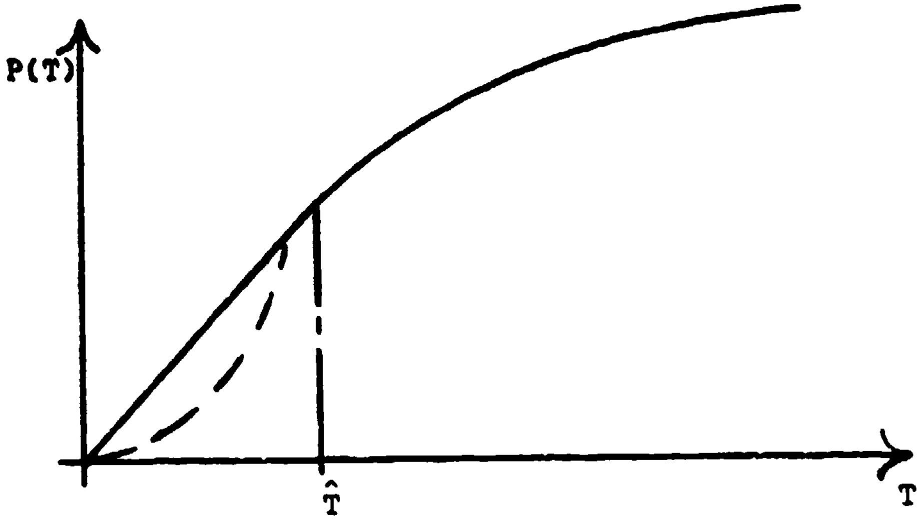


Figure 10. Minimal Concave Majorant

The point $\hat{T}$ is of some interest since the composite curve formed by the straight line from the origin to $(\hat{T}, P(\hat{T}))$ and the curve $P(T)$ for $T > \hat{T}$ is the minimal concave majorant for the entire function. $\hat{T}$ is determined by observing that the straight line of Figure 10 has the form

$$P(T) = P'(\hat{T})T , \tag{29}$$

in order that it be tangent to the curve $P(T)$ at $T = \hat{T}$. The solution of (29)then yields $\hat{T}$. In the event one has to resort to numerical techniques, a lower bound on $\hat{T}$ exists, namely, the upper bound on the region of convexity for the detection function , $\bar{T}$.

The modified optimization problem which yields the optimal policy for the original expected time problem is

$$\text{Max } P(T) = \sum_{i=1}^{N} \begin{Bmatrix} P'(\hat{T}_i)t_i, & t_i \leq \hat{T}_i \\ P(t_i), & t_i > \hat{T}_i \end{Bmatrix} \tag{30}$$

$$\text{S.T.} \sum_{i=1}^{N} t_i \leq T$$

$$t_i \geq 0 ,$$

for all $T > 0$.

The solution is obtained by applying the Kuhn-Tucker (K-T) conditions to the above concave programming problem. First the K-T conditions require that for the optimal solution, when $T < \text{Min } (\hat{T}_i)$ (where Ψ is the Kuhn-Tucker multiplier)

$$P_i(\hat{T}_i) = P'_i(t_i) = \psi$$

for all $t_i > 0$, and when

$$\min (\hat{T}_i) < T < \sum_{i=1}^{N} \hat{T}_i ,$$

$$P'_i(t_i) = \psi = P'_j(\hat{T}_j)$$

for some j for which $t_j < \hat{T}_i$. The last statement implies that once one starts allocating according to the FAR, say in the i^{th} region, one continues to allocate to that region until

$$P'_i(t_i) = P'_j(\hat{T}_j),$$

when $t_i \geq \hat{T}_i$. Then one begins to place *all* the additional time into the next region, j. This process is repeated

until that value of total available time, T, is attained such that the optimal policy has the property that

$$t_i > \hat{T}_i$$

for every i. Note this procedure is utilized to determine the allocation policy, the return for that policy is, of course, obtained by substituting the policy into the *original* problem.

To briefly summarize, we have discussed the procedure for solving the expected-time problem. Algorithmically,

1. Solve equation 29 for the $\hat{T}_i$, $i = 1,2,\ldots,N$.
2. Solve the maximization problem (equation 30) for all $T > 0$.
3. Using the results of Step (2), form

$$E = \int_0^\infty (1 - P(T))dT \ .$$

In the following paragraphs, we discuss the approach taken in each of the above steps. The solution of equation 29 requires that one solve a set of equations for the models of this chapter. First, the situation in which the distribution on the start of the visible period is uniform on (0,S) yields the following equation in $\hat{T}$, for the i^{th} region,

$$\hat{T}_i = \left\{\frac{S_i}{(e^{(\lambda_i+k_i)S_i}-1)e^{-(\lambda_i+k_i)\hat{T}_i}} - 1\right\}\frac{1}{\lambda_i + k_i} .$$

For the exponentially-distributed-start-time models, one obtains

(a) $(\lambda_i + k_i) \neq \beta_i$

$$\hat{T} = \frac{1}{\beta_i(\lambda_i+k_i)}\ [\beta_i-(\lambda_i+k_i)]e^{(\lambda_i+k_i)\hat{T}} + (\lambda_i+k_i) \cdot$$

$$(\beta_i\hat{T} + 1)e^{-(\beta_i-(\lambda_i+k_i)\hat{T}} - \beta_i$$

(b) $(\lambda_i + k_i) = \beta_i$

$$\hat{T} = \frac{1}{\beta_i}(e^{-\hat{T}} - \beta_i^2T^2 - 1)$$

Since the above expressions were derived from the pseudo-concave detection functions of this chapter we are assured of positive roots, other than $\hat{T} = 0$.

Having obtained the $\hat{T}_i$ for each region, we solve the resulting maximization problem using dynamic programming. The very nature of a dynamic programming solution allows us to obtain the optimal policy for all values of T up to some maximum, say T_{max}.[1]

Finally, the output from Step (2), the value of the probability of detection for each level of total effort using the

---

[1]Appendix E contains the program listings for these computations .

above policy, becomes the input to a numerical integration routine in order to compute the resulting expected time until detection.

There are two types of errors in this procedure: the first is the error in using numerical quadrature; and the second, possibly more important error, is that obtained from the restriction of the region of integration to the range $[0,T_{max}]$. Since expressions for the first type of error are readily obtained (Brand, 1960), we shall make some observations concerning the second error source. In Section 4.2, it was observed that the asymptotic form of the expression for the probability of detection was exponential in nature (See equation 22), the rate being a function of the search parameters. Given these facts, it seems reasonable to assume that one may readily determine an ξ such that

$$e^{-\xi T} \geq 1 - P(T) \text{ for } T > T_{max} .$$

Then the following relationship holds

$$\int_0^{\infty} e^{-\xi T}\, dT - \int_0^{T_{max}} e^{-\xi T}\, dT \geq \int_0^{\infty} (1 - P(T))\, dT - \int_0^{T_{max}} (1 - P(T))dT,$$

or

$$\frac{e^{-\xi T_m}}{\xi} \geq \int_0^{\infty} (1 - P(T))dT - \int_0^{T_{max}} (1- P(T))dT$$

Thus the above expression forms an upper bound on the second source of error in the numerical procedure. Of course, if we specify an upper bound for this source of error, say ε, then one may solve for the corresponding value of T_{max}, e.g.,

$$T_{max} > \frac{-1}{\xi} \ln (\varepsilon\xi) .$$

Finally, if one removes the assumptions made at the beginning of this section, i.e., that once the target appears it remains visible, then the expected time to detect the target increases without limit under the optimal policy. However, if we restrict our attention to the conditional expected time to detect, conditioned on detection occurring, then one can use the above procedure to determine the optimal allocation policy and the resulting value for the conditional expected time until detection.

4.3.2 Comparison with the Koopman Model

The procedure just outlined can also be utilized to determine how the Koopman allocation policies perform when minimizing the conditional expected time until detection. That is, one generates the values of the objective function for the problem of maximizing the probability of detection using the Koopman allocation policy for increasing levels of total available search time, then carries out the numerical integration required to compute the conditional expected time

until detection, conditioned upon detection ultimately occurring. Table 7 contains the results of a study of the sensitivity of the Koopman policy to changes in the detection rates for a 2-region search situation in which the

(a) searcher has the maximum uncertainty in the target's location ($p_1 = p_2$), and

(b) visibility parameters are identical.

Observe that the error (the ratio of the conditional expected times until detection) is minimized in the situation in which the detection rates are identical. Furthermore, when the detection rates are large relative to the visibility rates, the error is acceptable. Finally, the significant errors occur in the situation in which the detection rates are unequal[1].

4.4 General Single Interval of Visibility Model

4.4.1 Description

In this section we consider the situation in which the target may be either visible or masked with respect to the searcher whenever the latter enters the appropriate region. The probability vector describing these target states is

---

[1]The reader is referred to Figure 21, 5.3.2 for a comparison of the late-arrival version of the single interval model to the analogous multiple interval models. In addition, Section F.4 contains the maximum errors and their times of occurrence for the parameters of Table 7, as well as other combinations of parameters.

Table 7

The Sensitivity of the Conditional Expected Time Until Detection to the Detection Rates for the Random Interval Model

k_1 \ k_2	.1	1.
.1	1.3	1.9
1.	1.9	1.+

Ratio of Conditional Expected Times Until Detection

k_2 \ k_1	.1	1.
.1	28.9	30.1
1.	30.1	20.3

The conditional expected time until detection under the optimal policy, where $p_1 = p_2 = 1/2$, $\lambda_1 = \lambda_2 = 1.$, and $\beta_1 = \beta_2 = 0.1$.

$\overline{\Pi} = (\Pi, 1-\Pi)$, where Π is the visibility probability and $1-\Pi$ the masking probability. The vector $\overline{\Pi}$ is assumed known for each of the regions of interest. One might, for example, consider $\overline{\Pi}_j = (\Pi_j, 1 - \Pi_j)$ to be a vector representing environmental conditions affecting the searcher's detection gear. Physically, the single interval of visibility could represent the length of time required for the target to become aware of the searcher's presence. As an example, consider the scenario of a search for a submarine. The masked state is generated by the fact that the submarine may not have arrived on station at the time at which the search begins. The visible period could be generated by his length of time on station. The vector $\overline{\Pi}$ represents the searcher's prior probability on whether or not the target is on station yet; the vector P, the searcher's prior probability on target location.

In general, the target is visible with probability Π upon the entry of the searcher into the appropriate region. Given that the target is visible, then one has from Chapter 3 that the conditional probability of detection, given that t time units are spent searching, is

$$P(t) = \int_0^t \rho(\tau) A(\tau) d\tau \ .$$

On the other hand, given that the target is masked upon the searcher's entry, one has that

$$P(t) = \int_0^t f\,(u)M\,(t - u)du$$

where

$$M(t-u) = \int_0^{t-u} (s-u)\bar{H}(s-u)ds\,.$$

Then for the i^{th} region, we have that the probability of detection, given t time units spent searching, is

$$P_i(t) = P_i\,\Pi_i M_i(t) + (1 - \Pi_i)\int_0^t f_i(u)M_i(t - u)du \quad ,$$

$$= p_i\,\Pi_i \int_0^t \rho_i(\tau)H_i(\tau)d\tau + (1 - \Pi_i)\int_0^t f_i(u)M_i(t - u)du \qquad (31).$$

4.4.2 Model Solution

Let $\Pi = (\Pi_1, \ldots, \Pi_N)$ be a vector containing the probabilities that the target is visible at the start of the search in each region. We shall consider the example of Section 4.2 in which the lengths of visible periods and the times until the start of the same, given that the target is not visible with probability $1 - \Pi_i$, are exponentially distributed. For $\beta_i \neq (\lambda_i + k_i)$ the probability of detection under these assumptions is

$$P_i(t) = \Pi_i\left(\frac{p_i k_i}{k_i + \lambda_i}\right)\left\{1 - e^{-(\lambda_i + k_i)t}\right\}$$

$$+ (1 - \Pi_i)\left\{\frac{k_i p_i}{\lambda_i + k_i}\left[1 - \frac{[\beta_i e^{-(\lambda_i + k_i)t} - (\lambda_i + k_i)e^{-\beta_i t}]}{\beta_i - (\lambda_i + k_i)}\right]\right\}, \quad (32)$$

and for $\beta = \lambda_i + k_i$

$$P_i(t) = \Pi_i \frac{p_i k_i}{(k_i + \lambda_i)}\left\{1 - e^{-(\lambda_i + k_i)t}\right\}$$

$$+ (1 - \Pi_i)\left\{\frac{k_i p_i}{(\lambda_i + k_i)}\left[1 - e^{-\beta_i t}(1 + \beta_i t)\right]\right\}. \quad (33)$$

These detection functions are either strictly concave or pseudo-concave, i.e., convex until some point, $\bar{T}$, and concave thereafter. For equation 32 the convex region is

$$t_i \leq \frac{1}{\beta_i - (\lambda_i + k_i)} \ \ln\left\{\frac{(1 - \Pi_i)\beta_i^2}{(\lambda_i + k_i)[\beta_i - \Pi_i(\lambda_i + k_i)]}\right\}, \quad (34)$$

In the event the right-hand side is less than or equal to zero, the detection function is strictly concave. For equation 33, the condition for pseudo-concavity becomes

$$t_i \leq \frac{1 - 2\Pi_i}{\beta_i(1 - \Pi_i)} \quad . \quad (35)$$

The First Allocation Rule (FAR) for this model is:

Choose that region j for which

$$p_j k_j \Pi_j = \max_{1 \le i \le N} p_i k_i \Pi_i \quad .$$

Note that this is the Koopman FAR weighted by the probability that the target is visible. For small values of T, the rule becomes:

Choose j such that

$$p_j k_j \Pi_j + p_j k_j T(\beta_j(1 - \Pi_j) - \Pi_j(\lambda_j + k_j))$$
$$= \max_{1 \le i \le N} \left\{ p_i k_i \Pi_i + p_i k_i T(\beta_i(1 - \Pi_i) - \Pi_i(\lambda_i + k_i)) \right\} \quad .$$

If Π_i is viewed as the limiting probability of an alternating renewal process (the states being the visible and masked conditions of the target), then $\Pi_i = \left(\frac{\beta_i}{\beta_i + \lambda_i}\right)$.[1] In this case, it can be shown that the detection functions are concave.

Where the detection functions are concave, the switch time T^* (from region 1 to 2) is determined from the solution of

---

[1] See Section 5.2.1 for the proof of this assertion. The situation in which the visibility process is modeled as an alternating renewal process is treated in Chapter 5.

$$P_1(T^*) = p_2 k_2 \Pi_2 .$$

The extension of these ideas to N-regions basically involves the solution of the Kuhn-Tucker conditions for the N-region problem.

Again, we can specify an approximation for large values of T, the total available search time. Since this model is a weighted combination of the two previous ones, the conditions for an approximate solution are similar to those already given by equations 20-23. Consider the 2-region example of the previous section wherein $(\lambda_1 + k_1) > \beta_1$ and $\beta_2 > (\lambda_2 + k_2)$. The conditional detection functions are then approximated by

$$P_1(t_1) \simeq \frac{\Pi_1 k_1 p_1}{\lambda_1 + k_1} + (1 - \Pi_1)\left(1 - \frac{(\lambda_1 + k_1)e^{-\beta_1 t_1}}{[(\lambda_1 + k_1) - \beta_1]}\right)\frac{p_1 k_1}{k_1 + \lambda_1}$$

$$P_2(t_2) \simeq \frac{\Pi_2 k_2 p_2}{\lambda_2 + k_2}(1 - e^{-(\lambda_2 + k_2)t_2}) + \frac{(1-\Pi_2)p_2 k_2}{k_2 + \lambda_2}\left(1 - \frac{\beta_2 e^{-(\lambda_2 + k_2)t_2}}{\beta_2 - (\lambda_2 + k_2)}\right)$$

One obtains then for the approximate allocation scheme

$$t_1 = \frac{1}{\beta_1 + \lambda_2 + k_2}\left\{\ln[U_1/\xi_2] + (\lambda_2 + k_2)T\right\}, \qquad (36)$$

$$t_2 = T - t_1 ,$$

where

$$\xi_2 = p_2\Pi_2k_2 + \frac{(1 - \Pi_2)p_2k_2\beta_2}{[\beta_2 - (\lambda_2 + k_2)]} \text{ , and}$$

$$U_1 = \frac{(1 - \Pi_1)k_1p_1\beta_1}{[(\lambda_1 + k_1) - \beta_1]} .$$

4.4.3 Comparison with the Koopman Model

In this section, we examine the situation in which a partially informed searcher, being aware of the results of Koopman (which assume continuously visible targets), applies them to situations in which the target behavior is actually characterized by the random interval visibility process.

The different FAR, can lead to the erroneous selection of the initial region to receive the search effort, and the differing switch points can influence the shape of the error function. Figures 11 and 12 illustrate some of these remarks. Figure 11 contains the results of a parametric study of the effects of the prior probabilities on whether or not the target is visible (present) at the start of the search. The results indicate that the greatest errors in using the Koopman policy occur whenever it is highly probable that the target will be masked (or not present) at the start of the search. The limiting case of the single interval process is shown and represents the worst situation in terms of the use of the Koopman policy. Note that as the $\Pi_i \to 1$, the maximum percent

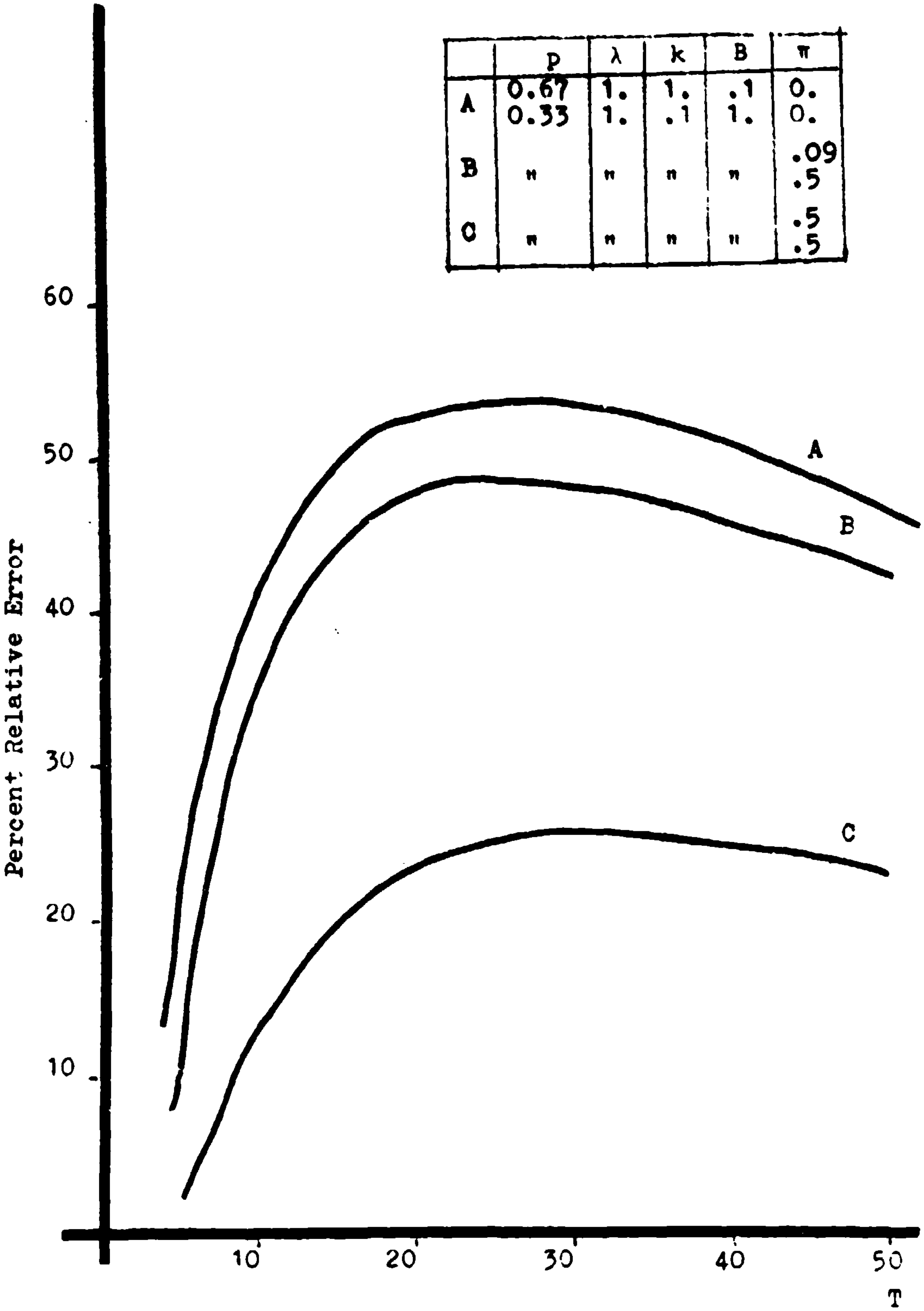


Figure 11 - Sensitivity to the visibility Parameters

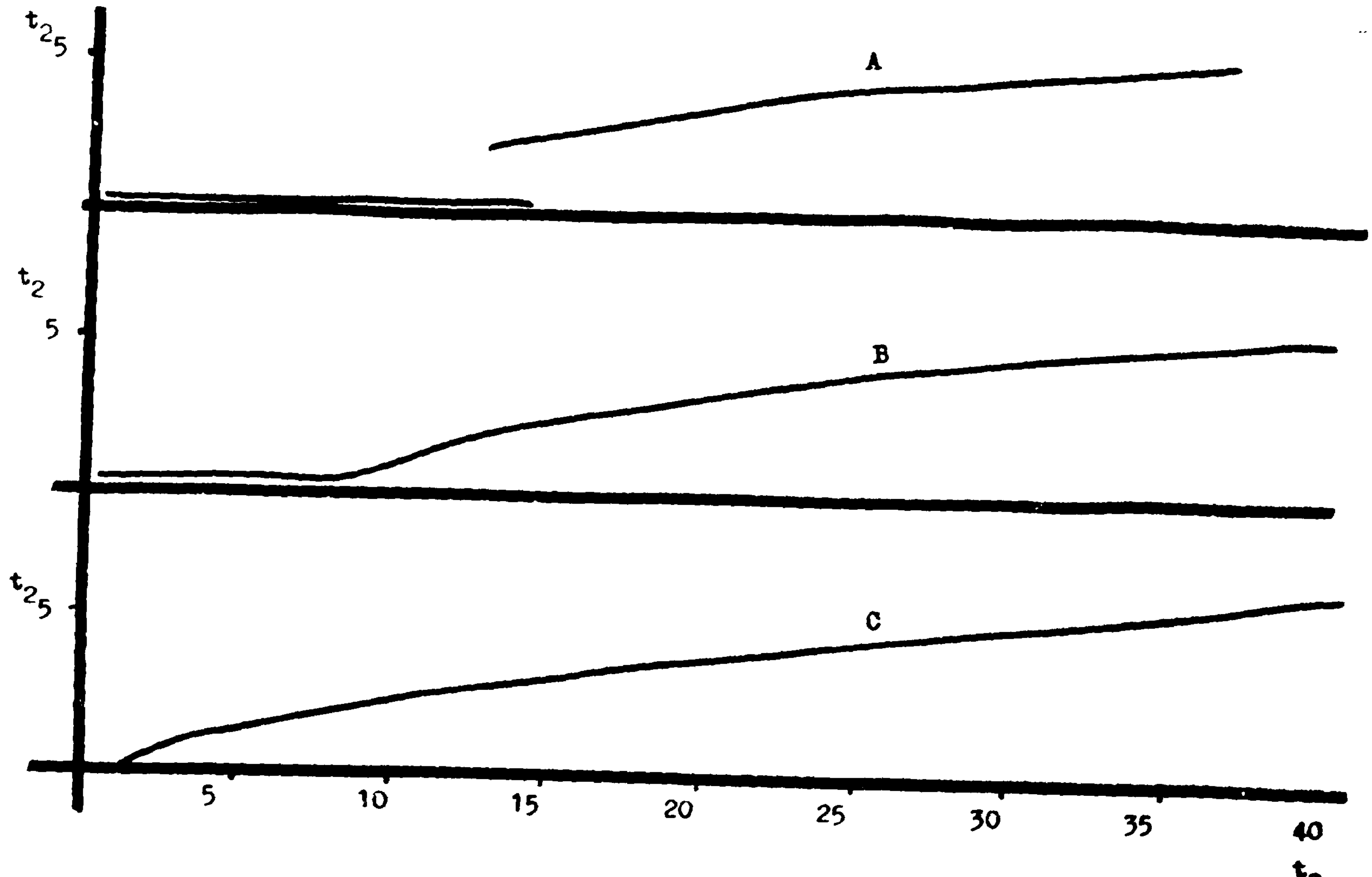


Figure 12- A Comparison of Allocation Policies

relative errors decrease, but the rate of decay of the error function is also decreasing. The decision maker who cannot estimate the visibility parameters will be concerned in such situations, since the lower rates imply that the option of increasing the available search effort to reduce the errors in using the Koopman policy may not be cost-effective.

Figure 12 displays the effects of variations in Π's on the optimal policies used in the determination of Figure 11. The higher the likelihood that the target is masked at the start of the search, the greater the length of time spent in region 1, for any fixed level of total available effort. Note that the Koopman policy (which is determined from k's), under large quantities of available search time, allocates the majority of the search time to the second region.

The approximate allocation policies given by equation 36 give rise to an approximate difference in the probability of detection under the two policies, optimal and the Koopman, as a function of time. For the conditions given on page 144, one obtains

$$E(T) = \frac{(1-\Pi_1)k_1 p_1}{\lambda_1 + k_1}\left\{\frac{\lambda_1 + k_1}{\lambda_1 + k_1 - \beta_1}\right\}\left\{e^{-\frac{\beta_1 k_2 T}{k_1+k_2}} - e^{-\frac{\beta_1(\lambda_2+k_2)T}{(\beta_1+\lambda_2+k_2)}}\right\}$$

$$+ \left\{e^{-\frac{(\lambda_2+k_2)k_1 T}{k_1+k_2}} - e^{-\frac{\beta_1(\lambda_2+k_2)T}{(\lambda_2+k_2+\beta_1)}}\right\}\left\{\frac{\Pi_2 k_2 p_2}{\lambda_2+k_2}\right.$$

$$\left. + \left[\frac{(1-\Pi_2)k_2 p_2}{(\lambda_2+k_2)}\right]\left[\frac{\beta_2}{\beta_2-(\lambda_2+k_2)}\right]\right\}.$$

We observe that the sensitivity results of this model are weighted combinations of the sensitivity results of Sections 3.2.3 and 4.2.3.

Appendix F contains the description and results of a numerical study of the error function. These are summarized below. It was noted that the Koopman policy *can* be used effectively in

(a) the homogeneous detector, homogeneous visibility parameter scenario, and

(b) situations in which the mean time to detect is much less than the mean length of the visible period.

The values of Π, the prior probability vector on target visibility at the start of the search, are of importance here. Since for $\Pi_i \to 1$, the general model reduces to the model of Chapter 3, which has, under the Koopman policy, errors when

T is small. If, on the other hand, $\Pi_i \to 0$, one has the random-interval, random-start-time model which may have significant errors when T is large under the Koopman policy.

Chapter 5

MULTIPLE PERIODS OF VISIBILITY

In this chapter we consider the situation in which the target may exhibit alternate periods of visibility and invisibility during the time which the searcher spends in the appropriate region. As noted in Chapter 1, this type of process can be used to characterize the behavior of a submarine on patrol within a region; a patrol moving through rugged terrain, foliage, etc.; a school of tuna or other fish. Initially, a simple Markovian visibility model is introduced for pedagogical purposes. Next the target's behavior is characterized as an alternating renewal process. The interaction between the target and the searcher is characterized as a Markov-renewal process. Having characterized the detection processes, we then study the problem of maximizing the probability of detection under a constraint on the available search time for certain special cases. Finally, the minimization of the expected time until detection, given unlimited search effort, is investigated.

5.1 *Description*

5.1.1 *A Simple Markovian Model of the Multiple Interval Process*

Assume that we are searching for a stationary target located in one of N regions with prior positional probability

vector P. Within the region containing the target, the visibility process is going on with transitions, according to the matrix

$$\begin{array}{l} \\ \text{Visible} \\ \text{Masked} \end{array} \begin{array}{c} \begin{array}{cc} \text{Visible} & \text{Masked} \end{array} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{array} ,$$

between the visible and masked states occurring at fixed, known time intervals. The presence of the searcher within the region in which the target is operating induces an absorbing state which is the detection state. The transition matrix for this new process is

$$\begin{array}{l} \\ \text{Visible} \\ \text{Masked} \\ \text{Detected} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Visible} & \text{Masked} & \text{Detected} \end{array} \\ \begin{bmatrix} 0 & 1 - p_{D_i} & p_{D_i} \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} ,$$

where p_{D_i} is the probability of detection within the i^{th} subregion.

Assume that a total of M glimpses are to be optimally allocated. We allocate m_j glimpses to region j, then seek the probability of absorption (detection). Let $f_{ij}^{(n)}$ denote the first passage probability from state i to state j, and X_n denote the state at the n^{th} transition, i.e.,

$$f_{ij}^{(n)} = \Pr\left\{X_n = j,\ X_m \neq j,\ m = 1, \ldots, n-1 \mid X_0 = i\right\} .$$

We can express $P_{ij}^{(n)}$, where

$$P_{ij}^{(n)} = \Pr\left\{X_n = j \mid X_0 = i\right\} ,$$

as

$$P_{ij}^{(n)} = \sum_{K=0}^{n} f_{ij}^{(n-K)} P_{jj}^{(K)} ,$$

where $f_{ij}^{(0)} = 0$. Let j denote the absorbing state. Then

$$\Pr\left\{\text{absorption} \mid n \text{ looks}, X_0 = i\right\} = \sum_{K=1}^{n} f_{ij}^{(K)} .$$

Consider the above expression and note that since j is an absorbing state, we have

$$P_{jj}^{(K)} = 1 ,$$

for K = 0, 1, 2, ..., hence

$$P_{ij}^{(n)} = \sum_{K=0}^{n} f_{ij}^{(n-K)} = \sum_{K=1}^{n} f_{ij}^{(K)} ,$$

since $f_{ij}^{(0)} = 0$.

From the above results we can formulate the following variation on Koopman's original problem.

$$\max \sum_{j=1}^{N} \Pr\left\{\begin{array}{c}\text{absorption (detection) in the} \\ j^{th} \text{ region}\end{array} \middle| m_j \text{ looks}\right\}$$

$$\text{S.T.} \sum_{j=1}^{N} m_j \leq M$$

$$m_j \geq 0 \ .$$

where

$$\Pr\left\{\begin{array}{c}\text{absorption (detection) in the} \\ j^{th} \text{ region}\end{array} \middle| m_j,\ X_0 = i\right\} = \sum_{K=1}^{m_j} f_{ij}^{(K)}$$

$$= P_{ij}^{(m_j)} \ .$$

The steady state probabilities of being in state i (the visible, and masked states of the visibility process) are (see Parzen (1962), p. 256),

$$(\Pi_1,\Pi_2) = (1/2,1/2) \ ,$$

therefore

$$\Pr\left\{\begin{array}{c}\text{absorption (detection) in the} \\ j^{th} \text{ region}\end{array} \middle| m_j\right\} = \sum_{j=1}^{2} P_{ij}^{(m_j)} \Pi_i p_j ,$$

where p_j is the probability that the target is in the j^{th} subregion.

For our simple process (for the i^{th} region)

$$P^1 = \begin{bmatrix} 0 & 1-p_D & p_D \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad P^2 = \begin{bmatrix} 1-p_D & 0 & p_D \\ 0 & 1-p_D & p_D \\ 0 & 0 & 1 \end{bmatrix}$$

$$P^3 = \begin{bmatrix} 0 & (1-p_D)^2 & p_D(1-p_D)+p_D \\ (1-p_D) & 0 & p_D \\ 0 & 0 & 1 \end{bmatrix},$$

$$P^4 = \begin{bmatrix} (1-p_D)^2 & 0 & p_D(1-p_D)+p_D \\ 0 & (1-p_D)^2 & p_D(1-p_D)+p_D \\ 0 & 0 & 1 \end{bmatrix}$$

$$P^5 = \begin{bmatrix} 0 & (1-p_D)^3 & p_D(1-p_D)^2+p_D(1-p_D)+p_D \\ (1-p_D)^2 & 0 & p_D(1-p_D)+p_D \\ 0 & 0 & 1 \end{bmatrix}$$

For ℓ even and $\ell > 1$, the entries in P^{ℓ} are

$$p_{23}^{(\ell)} = p_{13}^{(\ell)} = p_{13}^{(\ell-1)}$$

and for ℓ odd and $\ell > 1$, we have

$$p_{23}^{(\ell)} = p_{23}^{(\ell-1)}$$

$$p_{13}^{(\ell)} = p_D(1 - p_D)^{(\ell-[\frac{\ell}{2}+1])} + p_{13}^{(\ell-2)};$$

where [x] denotes the greatest integer in the term in brackets. The optimization problem becomes

$$\max \sum_{i=1}^{N} \sum_{K=i}^{2} p_{K3}^{(m_i)} \Pi_K \; p_i$$

$$\text{S.T.} \sum_{i=1}^{N} m_i \leq M$$

$$m_i \geq 0 \;, \quad i = 1, 2, \ldots, N \;,$$

where $p_{K3}^{(M_i)}$ is defined above. The above problem is readily solved using dynamic programming or the results of Wagner (1969) given in Appendix A.[1]

---

[1] A generalization of the transition matrix is

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}.$$

Then, given the presence of the searcher, the transition matrix for the search problem becomes

$$\begin{bmatrix} \alpha_i(1-p_D) & \beta_i(1-p_D) & p_D \\ \gamma & \delta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

5.1.2 Characterization of the Multiple Interval Search Scenario as a Semi-Markov Process

Disney (1970) characterized a visibility process in which the target alternates between visible and masked states as an alternating renewal process. The transition matrix for this process is

$$\begin{pmatrix} 0 & f_1(t) \\ f_2(t) & 0 \end{pmatrix}$$

where $f_1(t)$ is the probability density function for the time in the visible state and $f_2(t)$ the probability density function for the time in the masked state.

Using some renewal theory results, the author was able to obtain, among other things,

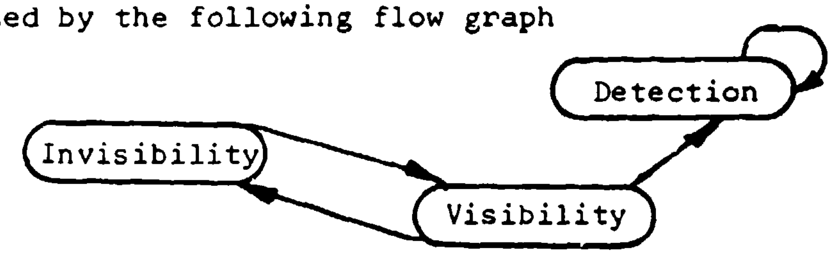
(a) $\Pi_1(t)$ the probability that the target is visible in $(t,t+dt)$.

(b) For a fixed time interval of length d, the distributions of

(1) the number of times the target is visible,

(2) the total time of visibility.

When the searcher enters the region in which the target is operating, we obtain the three-state process characterized by the following flow graph



The associated semi-Markov transition matrix is

$$Q = \begin{array}{c c} & \begin{array}{ccc} V & I & D \end{array} \\ \begin{array}{c} V \\ I \\ D \end{array} & \begin{pmatrix} 0 & R(t) & L(t) \\ F(t) & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array},$$

where

$R(t)$ = distribution function for the time spent in the visible state

$F(t)$ = distribution function for the time in the masked state

$L(t)$ = distribution function for the time until detection

$$l(t) = \frac{dL(t)}{dt}$$

$$r(t) = \frac{dR(t)}{dt} .$$

Next we derive the expressions for $l(t)$ and $r(t)$. Let $\rho(\tau)$ denote the conditional detection function, i.e., if τ_d is the time until detection, then

$$\rho(\tau)d\tau = \Pr\left\{\tau \le \tau_d \le \tau + d\tau \,\middle|\, \begin{array}{l}\text{target present and visible} \\ \text{for } \tau \text{ or more time units}\end{array}\right\} .$$

Let $h(t)$ be the probability density function for t_v, the length of the visible period.

Assuming that the process just entered the visible state, in order for the target to be detected in t time units, t_d must be in the interval $(t,t+dt)$ and t_v must have been at least equal to t, i.e.,[1]

$$\ell(t) = \rho(t)\bar{H}(t), \text{ where } \bar{H}(t) = \int_t^{\infty} h(t)dt .$$

If the target is to re-enter the masked state in t time units, then t_d must have been greater than t and t_v must equal t, i.e.,

$$v(t) = \bar{P}(t)h(t), \ \bar{P}(t) = \int_t^{\infty} \rho(\tau)dt .$$

The density function for the length of the masked period is $f(t)$.

Suppose the search must be carried out under a constraint on the total available search time, T. The problem of determining the optimal policy to maximize the probability of detection is

$$\max \sum_{i=1}^{N} \Pr \left\{ \begin{matrix} \text{Time until absorption in} \\ \text{region (i)} \le t_i \end{matrix} \right\}$$

$$\text{S.T.} \sum_{i=1}^{N} t_i \le T$$

$$t_i \ge 0 .$$

[1]Looking at the process just after a transition, for the situation in which one considers the searcher entering at an arbitrary time, the process becomes a delayed Markov renewal process with the relevant p.d.f. being the forward recurrance times of the visibility and invisibility processes.

We now introduce the distribution function for the first passage times as well as the marginal distribution function of the semi-Markov process. Let $G_{ij}(t)$ denote the distribution function for the first passage time, i.e.,

$$G_{ij}(t) = Pr[N_j(t) > 0 | X_0 = i] \equiv Pr[t_{j_1} \leq t | X_0 = i] ,$$

where t_{j_1} denotes the time of the first passage to state j and $N_j(t)$ denotes the number of entries to state j in time t. Let $P_{ij}(t)$ be defined as

$$P_{ij}(t) = Pr\{X_t = j | X_0 = i\} .$$

Appendix G contains a number of theorems due to Pyke (1961) which assist us in relating P and G. From Theorem 1 of Appendix G, we have[1]

$$P_{ij}(t) = P_{jj}(t)*G_{ij}(t) + \delta_{ij}(1 - H_i(t)) .$$

Let j denote the detection state, an absorbing state, i.e., $P_{jj}(t) = 1$, for $t \geq 0$. Since the search process will never start in state j, $\delta_{ij} = 0$, and from Theorem 1 of Appendix G, we have

$$P_{ij}(t) = 1*G_{ij}(t) = G_{ij}(t) .$$

The procedure for solving the optimization problem follows directly from the above results:

(1) By Theorem 2 we may determine $\Pi_{ij}(S)$ for the visible and masked states,

[1]See Appendix G for notation.

(2) inversion of $\Pi_{ij}(S)$ will yield $dG_{ij}(t)$ by virtue of the previous comments,

(3) having obtained $dG_{ij}(t)$ we are ready to study the properties of optimization problem.

The optimization problem can be restated in terms of the G functions as

$$\max \sum_{i=1}^{N} G_i(t_i)$$

$$\text{S.T.} \sum_{i=1}^{N} t_i \leq T$$

where $G_j(t) = p_j \cdot \sum_{\substack{\ell=1 \\ j\neq\ell}}^{2} G_{\ell j}(t)\Pi_\ell$, $\Pi_\ell = \Pr\{X_0=\ell\}$ and p_j is the probability that the target is in the j^{th} region.

Next we consider the following special case of the multiple interval process. Let the conditional detection density function be of the form

$$\rho(t) = ke^{-kt} ,$$

and the density functions for the lengths of the visible and masked times be, respectively[1]

[1]Bonder's (1970) discussion with the British on combat data indicated that this form for these processes appears to have an experimental basis.

$$h(t) = \lambda e^{-\lambda t}, \; f(t) = \beta e^{-\beta t} .$$

From the above results the density function for the transition time from the visible to the detected state is given by

$$\ell(t) = \rho(t)\bar{H}(t) = ke^{-(\lambda+k)t} .$$

Likewise, the density function for the transition time from the visible to the masked state is determined as

$$r(t) = p(t)h(t) = \lambda e^{-(\lambda+k)t} .$$

The following list gives the Laplace transforms for each of the above density functions:

Density Function	Laplace Transform
$g(t)$	$g(S) = \frac{k}{k + S}$
$h(t)$	$h(S) = \frac{\lambda}{\lambda + S}$
$f(t)$	$f(S) = \frac{\beta}{\beta + S}$
$\ell(t)$	$\ell(S) = \frac{k}{\lambda + k + S}$
$r(t)$	$r(S) = \frac{\lambda}{\lambda + k + S}$

The Laplace transform of the semi-Markov matrix $Q(t)$ is

$$q(S) = \begin{pmatrix} 0 & \frac{\lambda}{\lambda + k + S} & \frac{k}{\lambda + k + S} \\ \frac{\beta}{\beta + S} & 0 & 0 \\ C & 0 & e^{-S} \end{pmatrix},$$

and

$$I - q(S) = \begin{pmatrix} 1 & \frac{-\lambda}{\lambda + k + S} & \frac{-k}{\lambda + \quad + S} \\ \frac{-\beta}{\beta + S} & 1 & 0 \\ 0 & 0 & 1 - e^{-S} \end{pmatrix}$$

Defining the matrix $\Lambda(t)$ as follows

$$\Lambda(t) = (\delta_{ij}\Lambda_i), \quad \Lambda_i(t) = \sum_{j=1}^{3} Q_{ij}(t) .$$

Then

$$d\Lambda_i(t) = \sum_{j=1}^{3} dQ_{ij}(t)$$

and from the convolution property of the Laplace transform

$$\Lambda(S) = \delta_{ij}\mathcal{L}(d\Lambda_i(t)) = \prod_{j=1}^{3}\mathcal{L}(dQ_{ij}(t)),$$

or

$$\Lambda(S) = \begin{bmatrix} \frac{\lambda k}{(\lambda + k + S)^2} & 0 & 0 \\ 0 & \frac{\beta}{\beta + S} & 0 \\ 0 & 0 & 1 - e^{-S} \end{bmatrix}$$

$$I - \Lambda(s) = \begin{bmatrix} 1 \dfrac{-\lambda k}{(\lambda + k + S)^2} & 0 & 0 \\ 0 & 1 - \dfrac{\beta}{\beta + S} & 0 \\ 0 & 0 & 1-e^{-S} \end{bmatrix}$$

Pyke's Theorem of Appendix G states that

$$\Pi(S) = (I - q)^{-1}(I - \Lambda) ,$$

where

$$\Pi(S) = \mathcal{L}(P(t)) .$$

First we compute the required inverse,

$$(I - q)^{-1} = \frac{1}{D} \begin{bmatrix} 1 - e^{-S} & \dfrac{\lambda(1 - e^{-S})}{\lambda + k + S} & \dfrac{k}{\lambda + k + S} \\ \dfrac{\beta(1 - e^{-S})}{\beta + S} & 1 - e^{-S} & \dfrac{\beta k}{(\beta + S)(\lambda + k + S)} \\ 0 & 0 & 1 - \dfrac{\beta k}{(\beta+S)(\lambda+k+S)} \end{bmatrix}$$

where

$$D = |(I - q)^{-1}| = (1 - e^{-S})\left(1 - \frac{\lambda\beta}{(\beta + S)(\lambda + k + S)}\right)$$

We are interested in determining Π, where

$$\Pi = (I - q)^{-1}(I - \Lambda(S)) ,$$

in particular Π_{13} and Π_{23}. Carrying out the indicated matrix multiplication,

$$\Pi_{13}(S) = (1 - e^{-S})\left(\frac{k}{\lambda + k + S}\right)\frac{1}{D}$$

$$= \frac{k(\beta + S)}{[(\beta + S)(\lambda + k + S) - \lambda\beta]}$$

$$\Pi_{23}(S) = (1 - e^{-S})\ \frac{\beta k}{(\beta + S)(\lambda + k + S)}\ \frac{1}{D}$$

$$= \frac{\beta k}{[(\beta + S)(\lambda + k + S) - \lambda\beta]}\ .$$

Next the mean times to absorption are obtained from (Theorem 3, Appendix G)

$$\mu = \lim_{S\to 0^+} \frac{1}{S}\,(1 - q),$$

where

$$q(S) = \int_0^\infty e^{-St}\, dG_{ij}(t) = \int_0^\infty e^{-St}\, dP_{ij}(t) = \Pi(S),$$

for $j = 3$ the absorbing state.[1]

---

[1] The μ_{i3} is determined from

$$\mu_{13} = \lim_{S\to 0^+} \frac{1}{S}\left(1 - \frac{k(\beta + S)}{[(\beta + S)(\lambda + k + S) - \lambda\beta]}\right), \text{ or}$$

$$\mu_{13} = \frac{\beta + \lambda}{\beta\ k}\ , \text{ and}$$

$$\mu_{23} = \lim_{S\to 0^+} \frac{1}{S}\left(1 - \frac{\beta k}{[(\beta + S)(\lambda + k + S) - \lambda\beta]}\right), \text{ or}$$

$$\mu_{23} = \frac{\lambda + \beta + k}{\beta k}$$

The inversion of $\Pi_{13}(S)$ and $\Pi_{23}(S)$ is accomplished by partial fraction expansion. Since the denominator is common to both terms, we can factor it into a product of linear terms. Let

$$D(S) - (\beta + S)(\lambda + k + S) - \lambda\beta = S^2 + (\lambda + k + \beta)S + \beta k ,$$

the discriminant is then

$$b^2 - 4AC = (k + \lambda + \beta)^2 - 4\beta k = \lambda^2 + 2\lambda\beta + (k - \beta)^2 + 2\lambda k > 0 ,$$

since $k, \lambda, \beta > 0$. Both roots are negative since $\left(b^2 - 4AC\right) < b^2$, $4AC > 0$, implies $\left[-b \pm \sqrt{b^2 - 4AC}\right] < 0$. Denote the roots by γ_1 and γ_2, where γ_2 is the larger. This is possible since $b^2 - 4AC \neq 0$ for $k, \lambda, \beta > 0$. Expand $\Pi_{13}(S)$ by partial fractions to obtain

$$\Pi_{13}(S) = \frac{k(\beta + S)}{(S + \gamma_1)(S + \gamma_2)} = \frac{\frac{k(\beta - \gamma_1)}{(\gamma_2 - \gamma_1)}}{S + \gamma_1} - \frac{\frac{k(\beta - \gamma_2)}{(\gamma_2 - \gamma_1)}}{S + \gamma_2} .$$

Inverting we obtain

$$dP_{13}(t) = \frac{k(\beta - \gamma_1)e^{-\gamma_1 t}}{(\gamma_2 - \gamma_1)} - \frac{k(\beta - \gamma_2)e^{-\gamma_2 t}}{(\gamma_2 - \gamma_1)}$$

$$= \frac{k}{\gamma_2 - \gamma_1}\left[(\beta - \gamma_1)e^{-\gamma_1 t} - (\beta - \gamma_2)e^{-\gamma_2 t}\right] .$$

Expanding $\Pi_{23}(S)$ in the same fashion, yields

$$\Pi_{23}(S) = \frac{\beta k}{(S + \gamma_1)(S + \gamma_2)} = \frac{\beta k}{\frac{\gamma_2 - \gamma_1}{S + \gamma_1}} - \frac{\beta k}{\frac{\gamma_2 - \gamma_1}{S + \gamma_2}}$$

and inverting

$$dP_{23}(t) = \frac{\beta\ k}{\gamma_2 - \gamma_1}\left(e^{-\gamma_1 t} - e^{-\gamma_2 t}\right).$$

Integrating with respect to time, the desired conditional detection probabilities are

$$P_{23}(t) = \frac{\beta k}{\gamma_2 - \gamma_1}\left[\frac{1}{\gamma_1}(1 - e^{-\gamma_1 t}) - \frac{1}{\gamma_2}(1 - e^{-\gamma_2 t})\right] ,$$

and

$$P_{13}(t) = \frac{k}{\gamma_2 - \gamma_1}\left[\frac{(\beta - \gamma_1)}{\gamma_1}\left(1 - e^{-\gamma_1 t}\right) - \frac{(\beta - \gamma_2)}{\gamma_2}\left(1 - e^{-\gamma_2 t}\right)\right].$$

The limiting detection probabilities are,

$$\lim_{t\to\infty} P_{13}(t) = \frac{k\beta}{\gamma_1\gamma_2} = \lim_{t\to\infty} P_{23}(t) .$$

Also note the following facts:

(1) $\lim_{t\to 0^+} P_{13}(t) = 0 = \lim_{t\to 0^+} P_{23}(t)$, and

(2) $\lim_{t\to\infty} P_{13}(t) = \lim_{t\to\infty} P_{23}(t) = \frac{k\beta}{\gamma_1\gamma_2} = 1,$

Since

$$\gamma_2 = \frac{b + \sqrt{b^2 - 4AC}}{2A}, \quad \gamma_1 = \frac{b - \sqrt{b^2 - 4AC}}{2A},$$

$$\gamma_1\gamma_2 = C/A .$$

But $D(s) = S^2 + (\lambda + k + \beta)S + k\beta$, hence

$$\gamma_1\gamma_2 = k\beta.$$

We still require the probabilities of being in the visible and masked states at the start of the search. For the derivation of these probabilities, assume that the visibility process, as characterized by Disney (1970), has been going on for some time before the searcher enters the region containing the target, i.e., that this process has attained steady state. Let $\Pi_1(t)$ denote the probability of being visible at time t and $\Pi_2(t)$ the probability of being masked at time t. Assuming the process starts in visible state, then from Disney's results

$$\Pi_1(t) = F_1^c(t) + \int_0^t m_y(\mu)F_1^c(t - \mu)d\mu$$

$$\Pi_2(t) = \int_0^t m_z(\mu)F_2^c(t - \mu)d\mu \quad ,$$

where:

$F_1^C(t)$ = complementary cumulative distribution for the length of the visible period

$F_2^C(t)$ = complementary cumulative distribution for the length of the masked period

y = denotes the times between the renewals of the visibility events

z = denotes the times between the renewals of the masking events

$m_y(t)$ = the renewal density function for the y process

$$m_y(t) = f_y(t) + \int_0^t m_y(t - \lambda) f_y(\lambda) d\lambda \ .$$

$m_z(t)$ = the renewal density function for the z process

Under the assumptions made about the density functions in this example

$$m_y(t) = \frac{\lambda\beta}{\lambda + \beta} [1 - e^{-(\lambda+\beta)t}] \ ,$$

$$m_z(t) = \frac{\lambda}{\lambda + \beta} [\beta + \lambda e^{-(\lambda+\beta)t}] \ ,$$

and

$$\Pi_1(t) = \frac{1}{\lambda + \beta} [\beta + \lambda e^{-(\lambda+\beta)t}]$$

$$\lim_{t\to\infty} \Pi_1(t) = \frac{\beta}{\lambda + \beta} = \Pi_1(\infty)$$

$$\Pi_2(t) = \frac{\lambda}{\lambda + \beta}\left[1 - e^{-(\lambda+\beta)t}\right]$$

$$\lim_{t\to\infty} \Pi_2(t) = \frac{\lambda}{\lambda + \beta} = \Pi_2(\infty) .$$

Since the visibility process is a continuous parameter Markov process for this special case, these results follow from the theory of such processes[1].

Now given that a specific region is being searched, the probability of detection as a function of the time spent searching is

$$P(t) = P_{13}(t)\Pi_1(\infty) + P_{23}(t)\Pi_2(\infty) .$$

By summing across regions, we obtain the objective function for the problem of maximizing the probability of detection.

5.2 *Allocation of Effort to Maximize the Probability of Detection*

5.2.1 *Model Solution*

In this section the target motion, within a region, from the masked to the visible state and back again

---

[1]From the solution of

$$\frac{dP(t)}{dt} = P(t)\begin{pmatrix} -\lambda & \lambda \\ \beta & -\beta \end{pmatrix}.$$

is described as an alternating renewal process. The interaction of the searcher and the target is then modeled as a semi-Markov process. As an example of such a situation, consider an alternating renewal process (whose associated probability density functions are both exponential). The problem of maximizing the probability of detection under a constraint on the total available errort is given by[1]

$$\max \sum_{i=1}^{N} p_i[\Pi_i P_{v_i}(t_i) + (1 - \Pi_i)P_{m_i}(t_i)]$$

$$\text{S.T.} \quad \sum_{i=1}^{N} t_i \leq T \quad ,$$
$$t_i \geq 0 \quad , \quad \forall i \quad ,$$

where Π_i denotes the probability that the target is visible and

$$P_{v_i}(t) = \frac{k_i}{\gamma_{i_1} - \gamma_{i_2}} \left[\frac{(\beta_i - \gamma_{i1})}{\gamma_{i1}}[1 - e^{-\gamma_{i1}t}] - \frac{(\beta_i - \gamma_{i2})}{\gamma_{i2}}[1 - e^{-\gamma_{i2}t}] \right] , \tag{37}$$

[1]Unless otherwise noted, the parameters are identical to those introduced in the previous sections.

and

$$P_{m_i}(t) = \frac{\beta_i k_i}{\gamma_{i2} - \gamma_{i1}} \left\{ \frac{1}{\gamma_{i1}} (1 - e^{-\gamma_{i1} t}) - \frac{1}{\gamma_{i2}} (1 - e^{-\gamma_{i2} t}) \right\}, \tag{38}$$

where

$$\gamma_{i2} = \frac{b + \sqrt{b^2 - 4c}}{2},$$

$$\gamma_{i1} = \frac{b - \sqrt{b^2 - 4c}}{2} \tag{39}$$

and

$$b = \lambda_i + k_i + \beta_i$$

$$c = \beta_i k_i .$$

All of the visibility models with the exception of the one just presented have the following property.

$$\lim_{T\to\infty} P_i(T) \to \phi < 1 .$$

We have seen that, for the multiple interval model,

$$\lim_{T\to\infty} P_i(T) \to 1 .$$

The detection function for any region is convex over the interval $(o,\bar{t})$ where

$$\bar{t} \leq \frac{1}{(\gamma_2 - \gamma_1)} \ln[\gamma_2(\beta - \Pi_2)/\gamma_1(\beta - \Pi\gamma_1)] \ , \tag{40}$$

and concave thereafter.

A necessary consequence of having a detection function which is convex on $(0,\bar{t})$ is that

$$\frac{\beta}{\beta + \lambda + k} \geq \Pi \ . \tag{41}$$

If Π is the limiting probability that the alternating renewal process is in the visible state, then

$$\Pi = \frac{\beta}{\beta + \lambda} \ ,$$

and the above condition doesn't hold, i.e., the detection function is concave over the entire range of search effort.

The first allocation rule for this model is:

Choose the region j such that

$$p_j k_j \Pi_j = \max_{1 \leq i \leq N} p_i k_i \Pi_i \ , \tag{42}$$

which is identical to that of the general single interval model.

Let Π be the limiting probability that the alternating renewal process is in the visible state, then the switch time T_s is obtained from the solution of

$$P_1'(T_s) = p_2 k_2 \Pi_2 .$$

The solution of the above equation can be greatly simplified by making the following observations:

(a) $\gamma_2 > \gamma_1$

(b) $$P_1'(T) = \frac{p_1 k_1}{(\gamma_2 - \gamma_1)} \left\{ e^{-\gamma_1 T}(\beta_1 - \gamma_1 \Pi_1) - e^{-\gamma_2 T}(\beta_1 - \gamma_2 \Pi_1) \right\}$$

(c) $(\beta_1 - \gamma_1 \Pi_1) > 0$

$(\beta_1 - \gamma_2 \Pi_1) < 0$

Now if $\gamma_2 \Pi_1 - \beta_1 >> \beta_1 - \gamma_1 \Pi_1$ or, equivalently,

$$\frac{k_1}{\lambda_1 + \beta_1} >> 1 \quad \text{then}$$

one would expect the γ_2 term to dominate and the switch times will be "short." On the other hand, if $\gamma_2 \Pi_1 - \beta_1 < \beta_1 - \gamma_1 \Pi_1$ or $\frac{k_1}{\lambda_1 + \beta_1} \leq 1$, then one expects the γ_1 term to dominate and "longer" switch times will occur.

By making use of the observation that $\gamma_2 > \gamma_1$, approximate solutions can be developed which are applicable when "large" amounts of total search effort[1] are available. Under the above assumptions the conditional detection function becomes

$$P_v(t) \approx \frac{k}{\gamma_2 - \gamma_1} \left\{ \frac{\beta - \gamma_1}{\gamma_1} [1 - e^{-\gamma_1 t}] - \frac{\beta - \gamma_2}{\gamma 2} \right\}$$

$$P_m(t) \approx \frac{\beta k}{\gamma_2 - \gamma_1} \left\{ \frac{1}{\gamma_1} (1 - e^{\gamma_1 t}) - \frac{1}{\gamma_2} \right\} .$$

These approximations to the conditional detection functions are of the same general form as those investigated in Appendix A. For the 2-region case, the optimal allocations become

$$\begin{aligned} t_1 &\approx \frac{1}{\gamma_{11} + \gamma_{21}} \left\{ \ln[\phi_1/\phi_2] + \gamma_{21}\, T \right\} , \\ t_2 &\approx \frac{1}{\gamma_{11} + \gamma_{21}} \left\{ \ln[\phi_2/\phi_1] + \gamma_{11}\, T \right\} , \end{aligned} \tag{43}$$

where for γ_{i1}, the i denotes the region, and

$$\phi_1 = \frac{p_1 k_1 [\beta_1 - \Pi_1 \gamma_{11}]}{\gamma_{12} - \gamma_{11}} , \qquad \phi_2 = \frac{p_2 k_2 [\beta_2 - \Pi_2 \gamma_{21}]}{\gamma_{22} - \gamma_{21}} .$$

and assuming $\phi_1 > \phi_2$ the switch point is given by $T^* \approx \frac{1}{\gamma_{11}} \ln\left[\frac{\phi_1}{\phi_2}\right]$ To facilitate the comparison of these results with those of the previous chapters, recall that the γ's are given by

---

[1]These results will also be applicable in the situation in which $\gamma_2 \gg \gamma_1$.

$$\gamma_{i1} = \frac{\lambda_i + \beta_i + k_i - \sqrt{(\lambda_i + \beta_i + k_i)^2 - 4\beta_i k_i}}{2} \qquad (44)$$

$$\gamma_{i2} = \frac{\lambda_i + \beta_i + k_i + \sqrt{(\lambda_i + \beta_i + k_i)^2 - 4\beta_i k_i}}{2} \; .$$

The techniques developed in Appendix A are used to obtain a solution to the N-region problem under the above assumptions. Consider first the expression under the radical in equation 44.

$$(\lambda + k + \beta)^2 - 4k\beta = (k - \beta)^2 + \lambda(\lambda + 2k + 2\beta) \; .$$

Since all the parameters are positive, we can minimize this expression by choosing $k = \beta$ and $\lambda << 1$. In which case, γ_1 and γ_2 are given by $\gamma_1 \approx k + \lambda/2 - \sqrt{k\lambda}$ and $\gamma_2 \approx k + \lambda/2 + \sqrt{k\lambda}$. Physically this corresponds to the situation in which the mean length of the visible period is much larger than the mean length of the masked period which equals the mean time to detect within a region. Since the γ_i's are approximately equal to the k's, the Koopman allocation policy will provide a good approximation for large quantities of search time. If one has $\lambda = \beta$ and $k << 1$, then $\gamma_1 \approx k/2$ and $\gamma_2 \approx 2\beta$. From the previous discussion on switch times, this situation could lead to some very long switch times.

The condition $\lambda = k = \beta$ leads to the following values for γ_1 and γ_2

$$\gamma_1 = \frac{k}{2}(3 - \sqrt{5}) \text{ and } \gamma_2 = \frac{k}{2}(3 + \sqrt{5}).$$

This situation will also yield long term allocation policies which will agree with the Koopman scheme. Finally, by requiring $k = \lambda$ and $\beta << 1$, γ_1 and γ_2 become

$$\gamma_1 \simeq \beta/2 \text{ and } \gamma_2 \simeq 2k.$$

Since in this case $k \neq \beta$, one might again expect the Koopman policy to be inadequate for long-term allocations.

It was noted earlier in this section that the choice of Π_i as the limiting visibility probability of the alternating renewal process lead to a concave conditional detection function. The reaminder of this section will deal with the extreme limits on Π_i, i.e., $\Pi = 1.0$ or 0, since these cases are, as will be shown, direct generalizations of the single interval models. In Section 4.4 only a single interval of visibility was considered, here the target may exhibit multiple intervals; however, it is either initially masked or visible. In the former case, the conditional detection function is

$$P_i(t) = \frac{k}{\gamma_2 - \gamma_1}\left\{\frac{\beta - \gamma_1}{\gamma_1}[1 - e^{-\gamma_1 t}] - \frac{(\beta - \gamma_2)}{\gamma_2}[1 - e^{-\gamma_2 t}]\right\},$$

note that

$$P_v(\infty) = 1.$$

It is also noted that this function is concave for all values of

T since $\frac{\beta}{\beta + \lambda + k} < 1.$

Since $\Pi_i = 1$ for every i, the FAR is identical to the Koopman FAR, i.e.,

$$p_i k_i = \max_j p_j k_j \ .$$

The approximate solutions for large quantities of effort for a 2-region problem are

$$t_1 = \frac{1}{\gamma_{11} + \gamma_{21}} \left\{ \ln \frac{p_1 k_1}{p_2 k_2} + \ln \left[\frac{(\beta_1 - \gamma_{11})}{(\beta_2 - \gamma_{21})} \left(\frac{\gamma_{22} - \gamma_{21}}{\gamma_{12} - \gamma_{11}} \right) \right] \right.$$

$$\left. + \gamma_{21} T \right\}$$

and $t_2 = T - t_1$.

This model can be interpreted as a direct generalization of the single-interval-start-at-time-zero model, i.e., a multiple interval, start at time zero model.

One may interpret the case $(\Pi = 0)$ as a direct generalization of the single-interval, random-start-time model. The conditional detection function for this case is given by

$$P_m(T) = \frac{\rho k}{\gamma_2 - \gamma_1} \left\{ \frac{1}{\gamma_1} (1 - e^{-\gamma_1 t}) - \frac{1}{\gamma_2} (1 - e^{-\gamma_2 t}) \right\} \ .$$

This function is pseudo-concave, the region of convexity is the interval $(0,\overline{T})$ where $\overline{T}$ is given by

$$\overline{T} = \frac{1}{\gamma_2 - \gamma_1} \ln [\gamma_2 / \gamma_1] \ .$$

The FAR for this model is given by

$$p_i k_i \beta_i = \max_{1 \le j \le N} p_j k_j \beta_j$$

which is the FAR for the single-interval random-start-time model. The approximate allocations for large quantities of total effort for a 2-region problem are

$$t_1 = \frac{1}{\gamma_{11} + \gamma_{21}} \left\{ \ln \frac{p_1 k_1 \beta_1}{p_2 k_2 \beta_2} + \ln \left[\frac{\gamma_{22} - \gamma_{21}}{\gamma_{21} - \gamma_{11}} \right] + \gamma_{21} T \right\}$$

and $t_2 = T - t_1$.

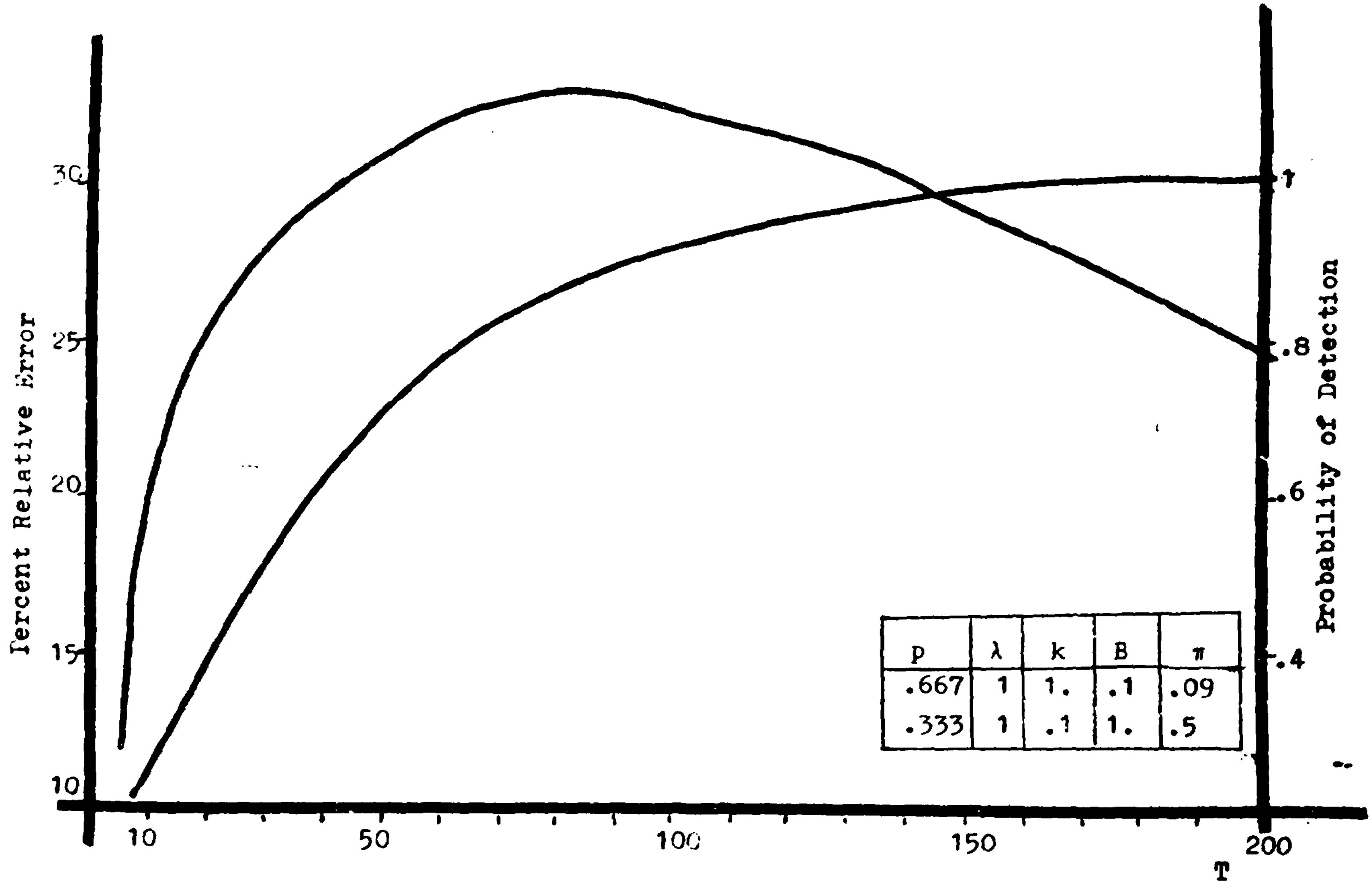
5.2.2 Comparison with the Koopman Model

In this section we examine the situation in which a partially informed searcher being aware of the results of Koopman (which assume continuously visible targets) applies them to situations in which the target behavior is actually characterized by the multiple interval intervisibility process[1].

[1]The reader is referred to Section 2.2.3 for a detailed discussion of the implications to the decision-maker (searcher) of such an analysis.

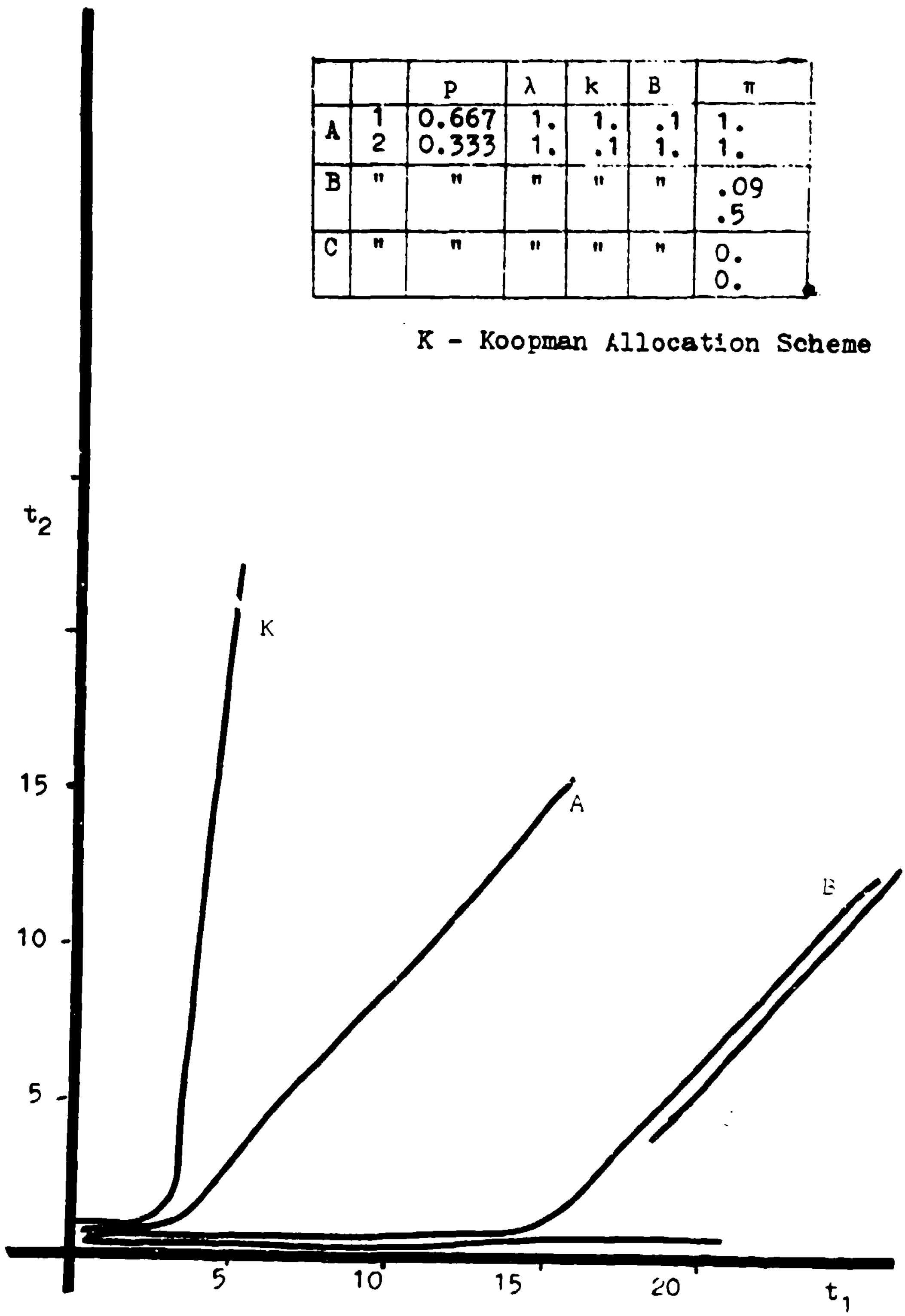
It is clear that, with the exception of the situation in which $\Pi = 1$, the FAR for this model could lead to the selection of a different initial region than that selected via the Koopman policy. It was also noted that obvious differences in switch points are possible.

Figure 13 is a composite plot of percent relative error versus available search time and the probability of detection under the optimal policy, versus available search time. We note that the peak percent relative error occurs when, under the optimal policy there is a large amount of available time i.e., the probability of de tion is 0.86. Even at T = 200, the relative error is still percent of the optimal return. Figure 14 compares the allocation policies, Koopman and optimal, utilized in obtaining the results shown in Figure 13. In addition, the effects of variations in the prior probabilities of target visibility on the allocation policies are shown. Finally, by way of comparison, an example in which the target arrives late but remains visible is included in Figure 14. (This is a special case of the random-single-interval model of Chapter 4). Observe that the sustained error function of Figure 13 results from the fact that the Koopman policy allocates the majority of any given level of search effort to Region 2, while the optimal solution does exactly the opposite (the approximately optimal solutions are given by equation 43). Figure 15 exhibits the probability of detection over a range of levels of available search time for each of the policies



p	λ	k	B	π
.667	1	1.	.1	.09
.333	1	.1	1.	.5

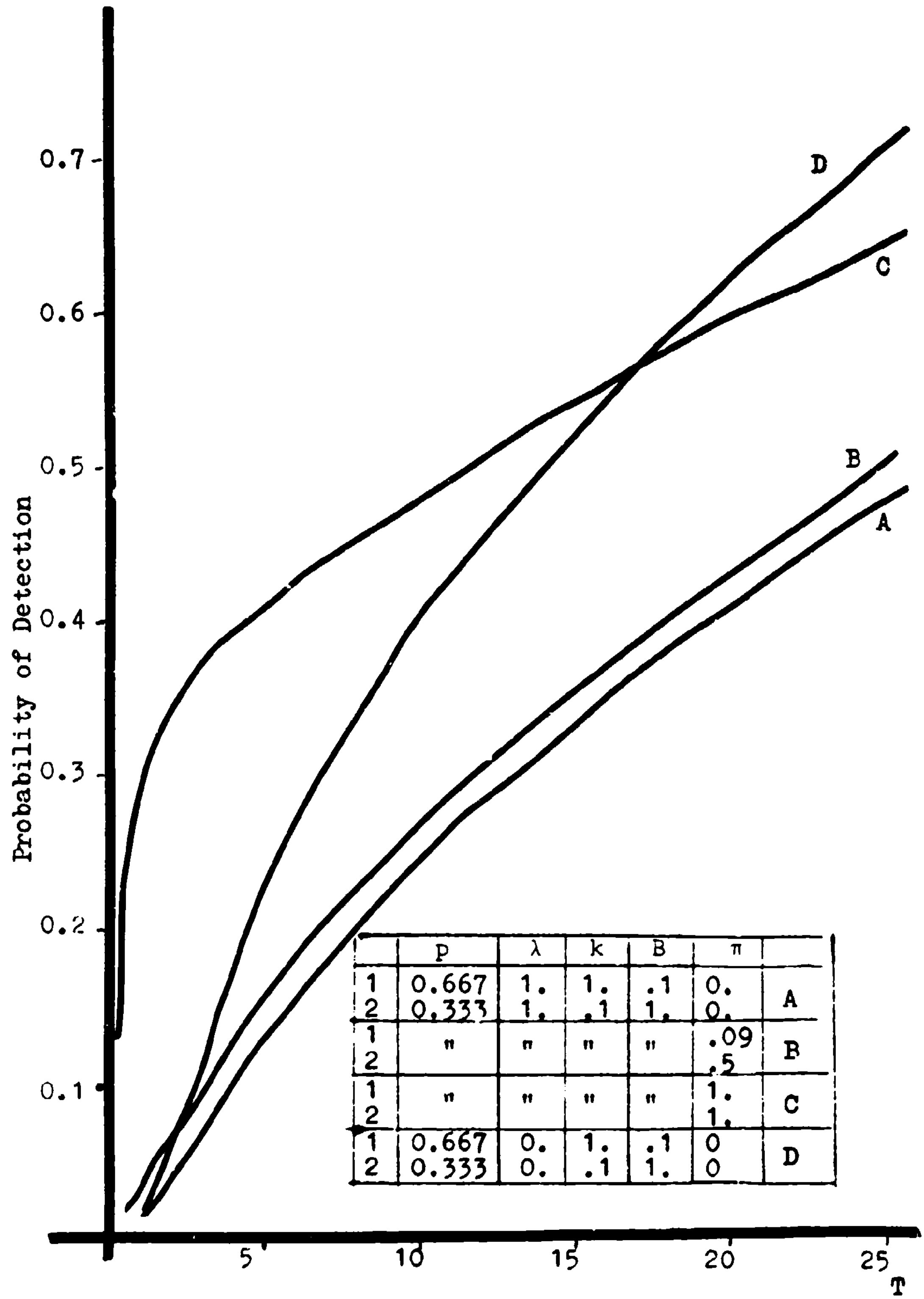
Figure 13 - Percent Relative Error and Optimal Return Versus Search Time



		p	λ	k	B	π
A	1 2	0.667 0.333	1. 1.	1. .1	.1 1.	1. 1.
B	"	"	"	"	"	.09 .5
C	"	"	"	"	"	0. 0.

K - Koopman Allocation Scheme

Figure 14 - Sensitivity of the Optimal Policy to Prior Estimates of Target Visibility



	p	λ	k	B	π	
1	0.667	1.	1.	.1	0.	A
2	0.333	1.	.1	1.	0.	
1	"	"	"	"	.09	B
2					.5	
1	"	"	"	"	1.	C
2					1.	
1	0.667	0.	1.	.1	0	D
2	0.333	0.	.1	1.	0	

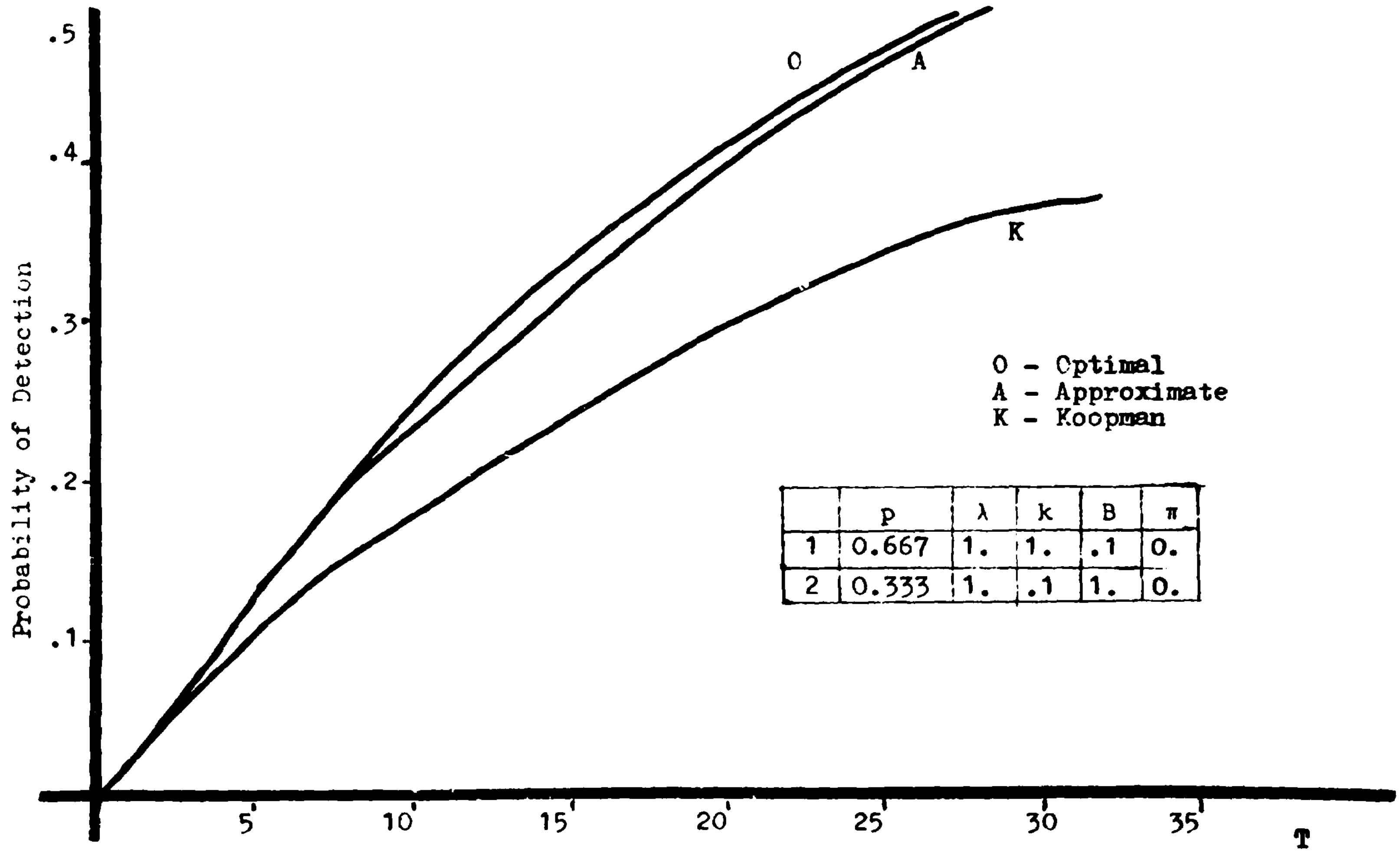
Figure 15 - Sensitivity of the Optimal Return To Prior Estimates of Target Visibility

described in Figure 14. The following observations are of interest:

(a) The assumption that the visibility probabilities are obtained from the limiting alternating renewal process results yields little difference in the optimal return when compared with the assumption that the target is initially masked. Thus the return, for these parameter levels, is insensitive to these assumptions.

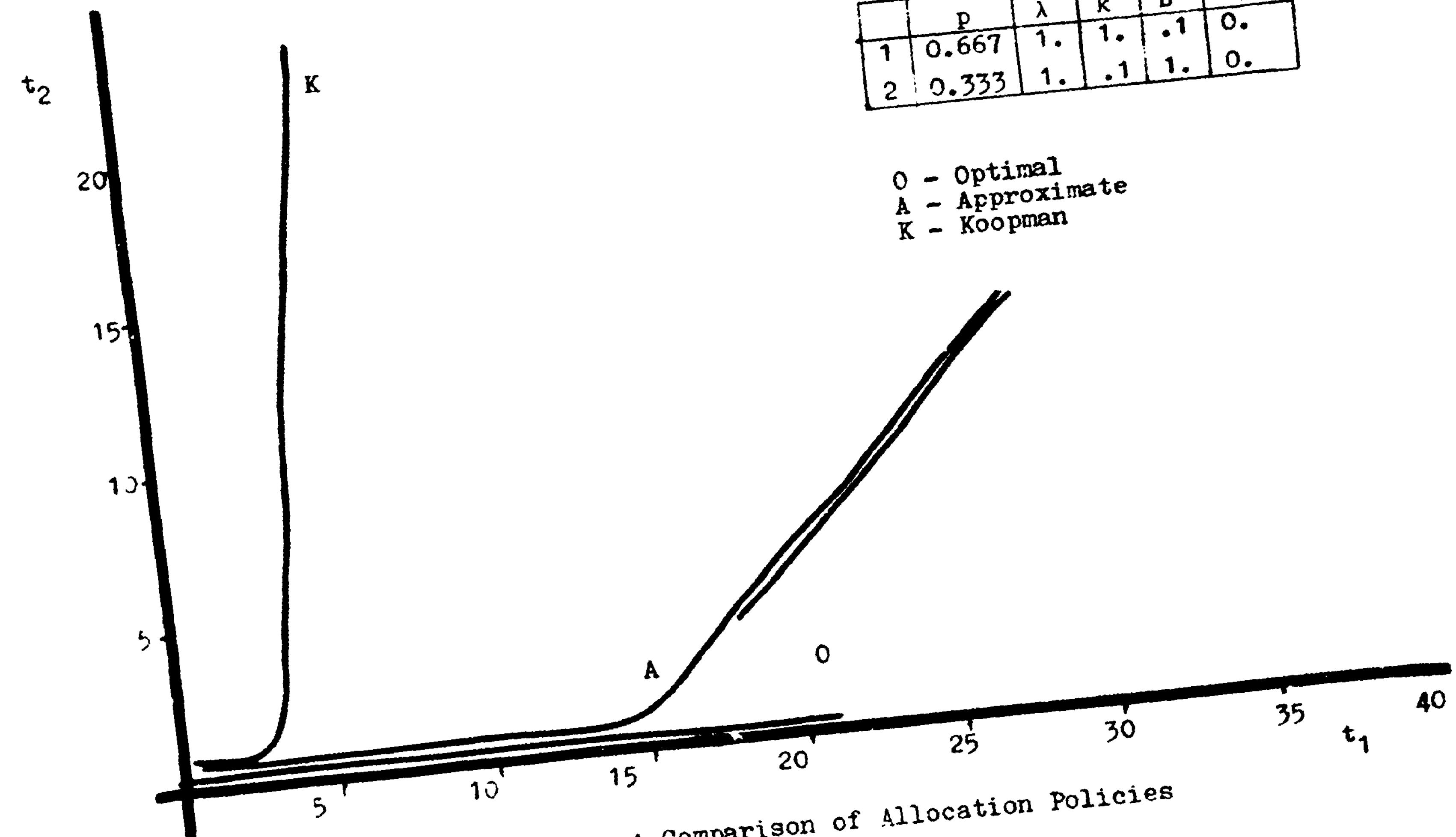
(b) Comparing the return under the assumption that the target is visible at the start of the search to the return in the late-arrival situation, one notes the rapid initial rise in the detection probability due to the initial visibility interval. However, since in the late arrival situation the target is always visible, the return in this case soon overtakes that for the multiple interval model. This indicates the advantage of intervisibility tactics. Also note that in comparing the late arrival situation to the multiple interval case in which the target is initially masked, the returns agree when the total available search effort is highly constrained. Thus in such situations the assumptions on target behavior are not critical.

Figure 16 compares the return (the probability of detection versus total available search time) from several allocation policies, the optimal policy, the approximate solution (equation 43), and the Koopman policy. In the example shown the approximate solution agrees very closely with the optimal solution over the entire range of available search time, the maximum error occurs in the vicinity of the switch point, $T=15$. Figure 17 gives the allocation policies associated with the



	p	λ	k	B	π
1	0.667	1.	1.	.1	0.
2	0.333	1.	.1	1.	0.

Figure 16 - A Comparison of the Results of Several Allocation Policies



	p	λ	k	B	π
1	0.667	1.	1.	.1	0.
2	0.333	1.	.1	1.	0.

Figure 17 - A Comparison of Allocation Policies

return functions of Figure 16. We note that the optimal policy is not strictly increasing with increasing available search time, while the approximate policy is. Figure 16 shows that the maximum error in using the approximate policy occurs for $15 \leq T \leq 20$, the two policies differ over the interval because the optimal policy is not strictly increasing.

Figure 18 displays the probability of detection under the optimal and Koopman policies. Note that in this case the peak difference occurs at a level of search effort which is far greater than the Koopman switch point ($T^* = 3$). The reader will recall that for the models of Chapters 2 and 3, the use of the Koopman policy lead to small errors for larger quantities of search time. Thus the earlier option of the decision-maker, to expend additional effort under the Koopman policy in order to reduce the errors in the probability of detection, is no longer cost-effective, i.e., an inordinate amount of additional effort would be required to achieve the desired reduction of the error function. Figure 19, a plot of the probability of detection under the optimal and Koopman policies, illustrates the points just made for the late-arrival, no-departure model.

From the earlier discussions of Section 5.2.2, one can construct approximate expressions for the long term error in using the Koopman policy in searching for an intervisible target. The two-region search situation leads to the following error expression

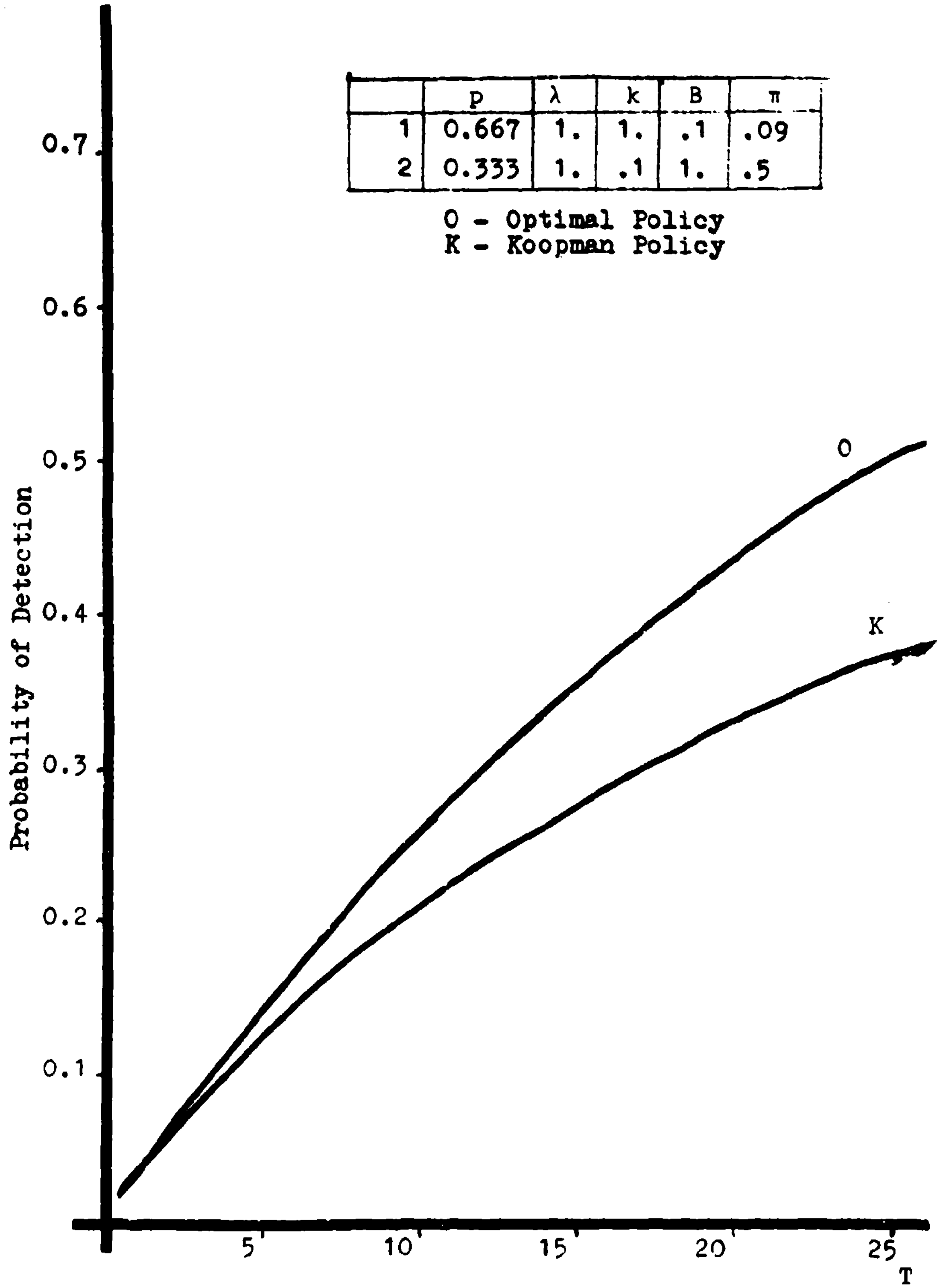
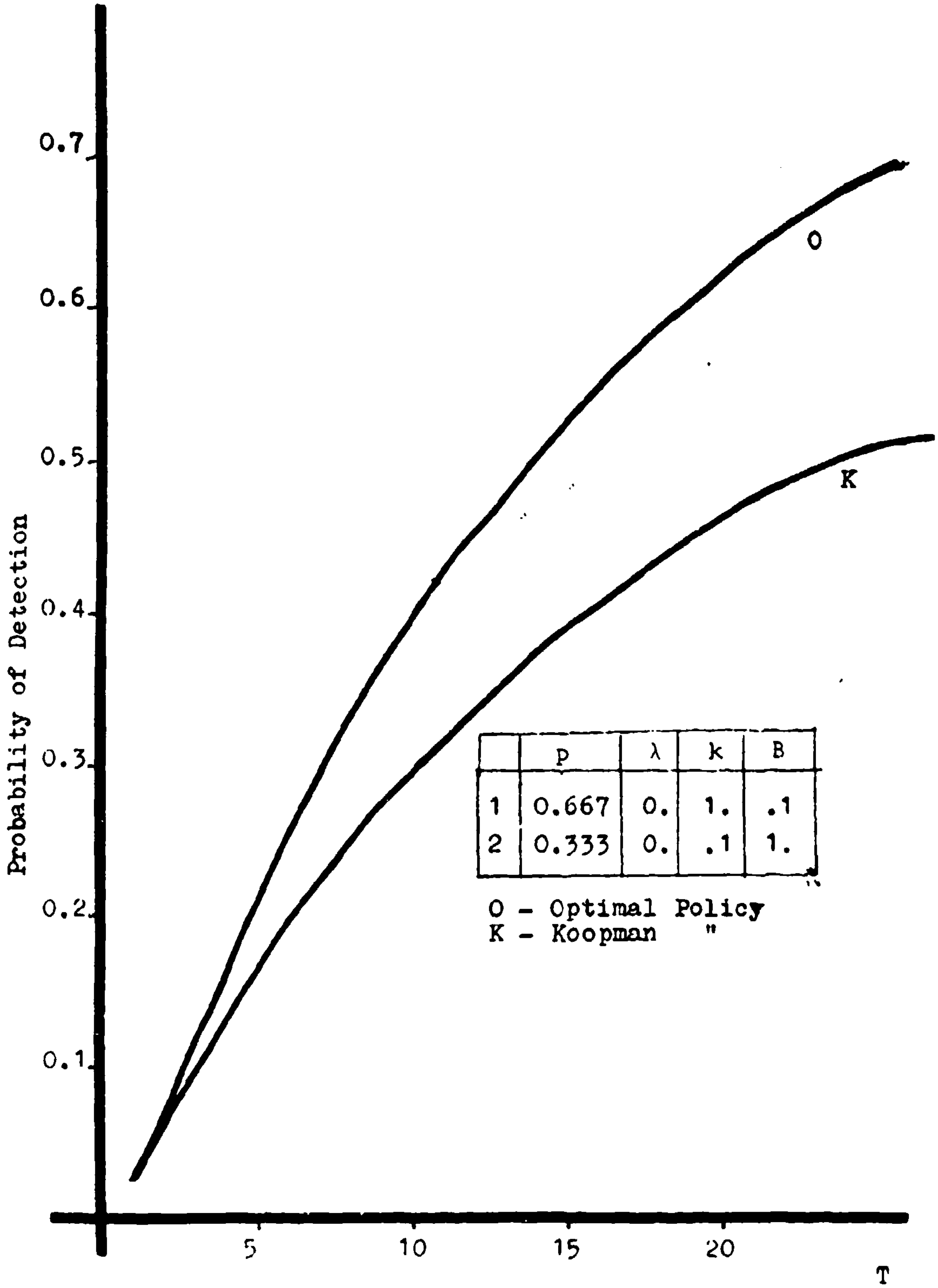


Figure 18 - A Comparison of Returns



	p	λ	k	B
1	0.667	0.	1.	.1
2	0.333	0.	.1	1.

Figure 19 - Probability of Detection versus Search Time

$$E(T) \approx \left(e^{-\frac{\gamma_{11}k_2T}{k_1+k_2}} - e^{-\frac{\gamma_{11}\gamma_{21}T}{\gamma_{11}+\gamma_{21}}}\right)\left\{\left(\frac{\beta_1-\gamma_{11}}{\gamma_{11}}\right)\left(\frac{p_1k_1\Pi_1}{\gamma_{12}-\gamma_{11}}\right)\right.$$

$$\left. + \frac{(1-\Pi_1)p_1k_1\beta_1}{(\gamma_{12}-\gamma_{11})\gamma_{11}}\right\}$$

$$+ \left(e^{-\frac{\gamma_{21}k_1T}{k_1+k_2}} - e^{-\frac{\gamma_{11}\gamma_{21}T}{\gamma_{11}+\gamma_{21}}}\right)\left\{\left(\frac{\beta_2-\gamma_{21}}{\gamma_{21}}\right)\left(\frac{p_2k_2\Pi_2}{\gamma_{22}-\gamma_{21}}\right)\right.$$

$$\left. + \frac{(1-\Pi_2)p_2\beta_2k_2}{(\gamma_{22}-\gamma_{21})\gamma_{21}}\right\} .$$

Again the knowledge of the above function provides answers to the following questions over a wide range of values of the parameters λ_i, k_i, and β_i.

(a) What is the sensitivity of the error function to changes in the parameters?

(b) How much time, T, must be available to insure that the error function is less than some specified value, E_0? (i.e., solve $E(T) = E_0$, for T).

5.2.3 Model Sensitivity

In the following paragraphs, we present the results of a sensitivity analysis with respect to the visibility parameters for both the optimal policy and its associated return. We shall be utilizing the approximate allocation policies and the resulting returns obtained in Section 5.2.2. Thus, it is implicitly assumed that sufficient search time is

available for the approximations to hold. The sensitivity of the allocation policy to changes in the visibility parameters is computed from

$$t_1 = \frac{\gamma_{21}T}{\gamma_{11} + \gamma_{21}} \quad \text{and} \quad t_2 = \frac{\gamma_{11}T}{\gamma_{11} + \gamma_{21}} \quad \text{as follows:}$$

$$\frac{\partial t_1}{\partial \lambda_1} = \frac{\partial t_1}{\partial \gamma_{11}} \frac{\partial \gamma_{11}}{\partial \lambda_1} ,$$

where

$$\frac{\partial t_1}{\partial \gamma_{11}} = - \frac{t_1}{\gamma_{11} + \gamma_{22}}$$

and

$$\frac{\partial \gamma_{11}}{\partial \lambda_1} = \frac{1}{2} \left(1 - \frac{(\lambda_1 + k_1 + \beta_1)}{(\lambda_1 + k_1 + \beta_1)^2 - 4\beta_1 k_1} \right) < 0 .$$

In the same fashion, one obtains

$$\frac{\partial t_1}{\partial \beta_1} = \frac{\partial t_1}{\partial \gamma_{11}} \frac{\partial \gamma_{11}}{\partial \beta_1} ,$$

where

$$-\frac{\partial \gamma_{11}}{\partial \beta_1} = \frac{1}{2} \left(1 - \frac{[(\lambda_1 + k_1 + \beta_1) - 4k_1]}{(\lambda_1 + k_1 + \beta_1)^2 - 4\beta_1 k_1} \right) > 0 .$$

One obtains for the sensitivity of t_1 to the visibility parameters of region 2,

$$\frac{\partial t_1}{\partial \lambda_2} = \left(-\frac{t_2}{\gamma_{11} + \gamma_{21}}\right)\frac{\partial \gamma_{21}}{\partial \lambda_2} \quad \text{and} \quad \frac{\partial t_1}{\partial \beta_2} = \left(\frac{t_2}{\gamma_{11} + \gamma_{21}}\right)\frac{\partial \gamma_{21}}{\partial \beta_2} .$$

The relative sensitivity of the λ's and β's for a given region is readily determined by the comparison of $\frac{\partial \gamma_{i1}}{\partial \lambda_i}$ and $\frac{\partial \gamma_{i1}}{\partial \beta_i}$, since the other terms are constant.

Considering the example of Figure 17, we have

$$\frac{\partial \gamma_{11}}{\partial \lambda_1} \doteq .025 \text{ and } \frac{\partial \gamma_{11}}{\partial \beta_1} \doteq 1 ,$$

indicating the dominance of the rate for the masked period at any appropriate level of total available search time. From the symmetry of the two-region search situation, the sensitivity calculations for t_2 are apparent.

The explicit computation of the partial derivatives of the optimal return with respect to the visibility parameters is prohibitive, e.g., one must compute

$$\frac{\partial P(T)}{\partial \lambda_1} = \frac{\partial P(T)}{\partial \gamma_{11}} \cdot \frac{\partial \gamma_{11}}{\partial \lambda_1} + \frac{\partial P(T)}{\partial \gamma_{12}} \cdot \frac{\partial \gamma_{12}}{\partial \lambda_1} \text{, etc.,}$$

However, the dynamic programming solution technique is utilized to obtain some numerical approximations via difference functions. Again, considering the example of Figure 17, we obtain the

following results for T = 30;

$$\frac{\partial P(T)}{\partial \lambda_1} = -0.14, \quad \frac{\partial P(T)}{\partial \beta_1} = 0.69, \quad \frac{\partial P(T)}{\partial \lambda_2} = -0.05, \text{ and } \frac{\partial P(T)}{\partial \beta_2} = 0.$$

Again we note the dominance of the rate for the masked period.

A numerical study of the error function is contained in Appendix F and is summarized below. The Koopman policy is adequate in the following situations:

(a) a search scenario characterized by identical detection rates and identical visibility parameters.

(b) whenever the mean time-to-detect is much less than the mean length of the visible periods.

On the other hand, in the search situations characterized by heterogeneous visibility conditions, the resultant approximate rates (γ_i's) may differ significantly from the detection rates used in the Koopman policy, resulting in error functions which continue to increase far beyond the Koopman expected time to detect. Recall that (a) and (b) also held in the General Single Interval Model of Section 4.2, when one used the limiting probabilities of an alternating renewal process as the probabilities of target visibility at the start of the search. In contrast, the Koopman policy was not adequate for the single interval model with random starting times in Cases (a) and (b).

5.3 *Minimization of the Expected Time Until Detection*

5.3.1 *Model Solution*

As was noted in Section 5.2, the forms of the detection functions for the examples therein ranged from pseudo-concave to concave depending upon the choice of the initial state probabilities for the masked and visible states. Regardless of the form of the detection function one has, for this model, the result that the limiting value of the probability of detection is unity. If the detection function is concave, Dobbie's (1963) results imply that the policy which maximizes the probability of detection for all levels of effort is also the policy which minimizes the expected time until detection, under unlimited available search time. On the other hand, if the initial state probabilities are such that the detection function is pseudo-concave, the results described in [4.3.1] are employed to solve the expected time problem.

Computationally then, one first utilizes the criteria of Section 5.2 to determine the form of the detection function. If it is pseudo-concave, Steps 1-3 of the procedure given in [4.3.1] are followed; if concave, the output from the dynamic programming solution to the constrained time problem is taken as the input to the numerical integration routine in order to determine the minimum expected time.

5.3.2 Comparison with the Standard Model

Again we determine the effectiveness of the Koopman allocation in this situation by generating the value of the probability of detection under the Koopman policy and carrying out the required numerical integrations. The approximate solutions discussed in Section 5.2 may also be utilized, as was the Koopman policy, in generating the expected time until detection in order to compare this result with the actual minimum. Figure 20 illustrates these points (See also Figures 15-19) and compares the multiple interval models with a late-arrival, no-departure version of the random interval models of Chapter 4. Note that, for the parameters of Figure 20, the assumptions that the prior probabilities on target visibility are the limiting results of an alternating renewal process, and that the target is masked at the start of the search, lead to essentially the same value for the expected time until detection. Hence the appropriateness of the Koopman policy is unaffected, in this case, by the difference in the prior probabilities on the target's visibility status at the start of the search. Figure 21 shows the sensitivity of the Koopman policy to variations in the detection rates. Since the visibility parameters λ and β are not identical over the regions of interest, one is not assured that under identical detection rates (of the order of the visibility rates) the Koopman policy will yield good results for the

expected time problem. Appendix F contains the maximum errors and their times of occurrence for the parameters of Figure 21, as well as other combinations of visibility parameters.

Ident. Symbol	Regions (2) R	Prior on-tgt. Loc. p	Visibility Rates λ	Detection Rates k	Masking Rates β	Prior on Visibility Π	Expt. Time (optimal)	Expt. Time (Koopman)	Model Desc.
	1 2	2/3 1/3	1 1	1. .1	.1 1.	0. 0.	* 39.4	88.2	Multiple Interval
×	1 2	"	"	"	"	0.09091 0.5	38.2	84.5	"
0	1 2	"	"	"	"	1. 1.	28.9	49.2	"
4	1 2	"	0. 0.	"	"	- -	* 20.4	42.3	Late Arrival No Departure

*Using the allocation policy generated by the minimal concave majorant of the detection functions.

All the values for the expected search time were obtained via numerical integration over a finite interval, the lengths of the intervals as well as the terminal values of the probability of detection under the various policies are given below

Ident. Symbol	T_{max}	$P_D(T_{max})$ Optimal	$P_D(T_{max})$ Koopman
	200	.99	.75
×	200	.99	.76
0	200	.99	.87
4	100	.99	.77

Figure 20 Sensitivity of the Expected Time Until Detection To the Prior Estimates On Target Visibility

k_2 \ k_1	.1	1.
.1	1.2	2.2
1.	1.+	1.3

Ratio of the Expected Times Until Detection

k_2 \ k_1	.1	1.
.1	108	38.2
.1	77.2	18.8

The expected time until detection under the optimal policy for

p	λ	β
0.667	1.	.1
0.333	1.	1.

Figure 21 The Sensitivity of the Expected Time Until Detection to the Detection Rates

Chapter 6

SUMMARY, CONCLUSIONS AND AREAS FOR FUTURE RESEARCH

In this chapter the important results for each of the search models presented in Chapters 2-5 are summarized and several areas for future research introduced. The results are presented in the order in which the models were described. The areas for future research are then presented.

6.1 Summary and Conclusions

6.1.1 The Binary Visibility Model

In the study of the Binary Visibility model of Chapter 2, the concept of the First Allocation Rule (FAR) was introduced and the FAR for this model was shown to differ from that of the classical Koopman results, i.e., the use of the Koopman FAR could lead to incorrect selection of the initial region to be searched. It was noted that the switch points in the 2-region search situation may differ from those of the classical search problem in either a positive or negative direction. In a situation in which the visibility probabilities are identical across the N-regions (not necessarily equal to one), the optimal solutions are identical to those obtained from classical search theory in which the target is assumed to be continuously visible. In comparing the optimal policies, Binary versus Koopman,

it was observed that the long-term allocation rates are identical. Thus the errors, when one uses the classical results in this situation, are maximized in situations in which the visibility conditions are heterogeneous and the available search effort is highly constrained. Hence, a searcher may compensate for a lack of knowledge of the visibility parameters at the cost of increasing the amount of effort allocated to the search.

Sensitivity analyses indicated that the optimal *policy* is robust with respect to changes in the visibility parameters whenever the detection rates are large. Sensitivity of the optimal policy to changes in the visibility parameters is independent of the level of available search effort.

Explicit expressions are provided for the optimal policies in the discrete detection and visibility process situation and it is shown that when one equates the infinitesimal detection and visibility probabilities to their discrete counterparts, the optimal allocations are approximately equal.

In considering the problem of minimizing the expected time until detection, one is forced to restrict attention to the conditional version of this problem, since the ultimate probability of detection is less than unity. Although there are many situations in which the use of the Koopman policy is approximately optimal in maximizing the

probability of detection, it may be inefficient when used in the expected time problem in a search situation in which the target is not continuously visible.

6.1.2 Random Interval of Visibility: Initially Visible

In this case the First Allocation Rule is identical to the classical results; however, the switch point for this model is always less than that of the classical search situation, i.e., one always begins to allocate some effort to the second region at a lower level of total available search time. Viewing this model as representing a search for a target which has its own detection capabilities, it is shown that wherever the searcher and target have identical regional capabilities (not necessarily equal), and the state of maximum prior uncertainty on the target position exists, the Koopman or classical policy is optimal. Hence, if the searcher is willing to make this assumption, or has data to verify these conditions, he may proceed as though he were searching for a continuously visible target. Furthermore, the assumption of maximum uncertainty in the prior location was later shown in specific numerical examples not to be critical. These results are valid for the N-region search situation.

In comparing optimal policies and returns, classical versus the Random Interval, it was noted that the peak relative error may occur at a level of search time approximately equal to the expected time to detect the target (denoted KET) under the optimal policy in situations in which the target is continuously visible. In such situations, one may expect relative errors of 10 and 5 percent at levels of search effort equal to two and three times the KET, respectively. Thus the searchers option of increasing the search time to offset a lack of knowledge of the visibility parameters may (depending on the acceptable error) no longer be cost-effective. These situations can occur whenever the visibility conditions are regionally heterogeneous.

The use of the classical policies in the problem of minimizing the conditional expected time until detection can lead to large errors even though the classical policy is approximately optimal in the problem of maximizing the probability of detection. It was observed in this context that, if the target's detection rates are greater than those of the searcher, it is important that the searcher obtain estimates of these in order to conduct an effective search.

Sensitivity analyses indicated that, in contrast to the results for the Binary model, the sensitivity of the

optimal *policy* to variations in the visibility parameters increases with increasing available search effort; however, the regional change as a percentage of the total allocation to that region remains constant. The sensitivity of the optimal allocation in the first region to variations in the visibility parameters of the second region is proportional to the current level of search effort in the second region. It was noted that the limiting sensitivity, as $T \to \infty$, of the optimal *return* to variations in the visibility parameters of a given region is proportional to the ultimate probability of detection in the region, where the constant of proportionality is the conditional mean time until detection in that region (conditioned on detection occurring).

The situation in which the detection and visibility processes are discrete was modeled and the optimal policies explicitly obtained. The optimal policies for the discrete and continuous detection models are shown to be approximately equal whenever the infinitesimal detection and visibility probabilities are approximately equal to their discrete analogs.

6.1.3 *Random Interval of Visibility: Random Initiation and Limit*

Application of the classical FAR in this situation could lead (in many cases) to the erroneous selection of the initial region to be searched. The model gave rise to pseudo-concave conditional detection functions, thus presenting the first "practical" examples of this more general class of detection functions. The optimal allocation policies resulting from such models have the property that they are not strictly non-decreasing with increasing total available search effort. Such policies have not previously appeared in the open literature on search and reconnaissance. Furthermore, in the case of identical regions it is shown that one doesn't start the allocation procedure by dividing the effort equally among the identical regions. The nature of these optimal policies implies that in the random-interval, random-start time model, one must have a minimum level of available time (which can be numerically determined from the process parameters) before beginning to search a new region. In the event that level of total available search time is insufficient in this respect, the optimal policy is to continue placing all the effort in one region rather than a small quantity in a new region. Recall that in this model that target is not visible (or present) at the time the searcher arrives in the

appropriate region, but is to become visible (or arrive) after some random length of time. Hence the return from placing small additional amounts of effort into a region in which one has been searching is greater than that obtained from allocating the same amount to a second, previously unsearched region, provided the total available search time is less than the sum of the minimum levels of search time for each of the two regions.

The pseudo-concave character of the conditional detection functions precludes the derivation of explicit expressions for the optimal policies. However, approximate policies are developed and the requisite conditions for their applicability discussed. It is shown that these policies are determined from the lesser of the two rates associated with the regional visibility processes--the first being the rate associated with the length of the masked period (β); the second, the conditional detection rate ($\lambda + k$) conditioned on detection occurring in the visibility interval. One can easily determine the conditions under which the classical optimal allocation policies will coincide with the approximate policies, or will greatly differ from them. Such observations are directly applicable to N-region search situations. Thus, in contrast to the models of Chapters 2 and 3, it is shown that

(a) the use of the Koopman policy will not be adequate in situations in which the visibility conditions are homogeneous across the regions of interest,

(b) the error function is not reduced by ir asing the available search time to some real: level,

(c) the maximum errors do not occur in situations in which the total available search time is highly constrained, and

(d) the availability of extremely good detectors (high rates) will not imply that the Koopman (or classical) policy yields small errors.

These results suggest that the searcher, in general, must obtain accurate estimates of the visibility parameters in order to conduct an effective search.

The model was shown to specialize to a late-arrival, no-departure model (by choosing $\lambda = 0$) and all of the analyses are directly applicable. It, in turn, was shown to be a special case of what was termed the General Single Interval model. This model has detection functions ranging from concave to pseudo-concave depending upon the levels of the a priori probabilities of initial target visibility, Π. The approximate policies just discussed were shown to be applicable here also.

Numerical studies comparing the classical and optimal search policies indicated that the classical policy, while not necessarily optimal, could be used effectively for the

General Random Interval model in situations in which (a) the mean time required to detect the target is much less than the mean length of the visible period or (b) the detection and visibility rates are identical across regions.

Sensitivity analyses indicated that the sensitivity of the optimal policy to changes in the visibility process parameters increases with increasing total available search effort; however, the change as a percent of the total allocation remains constant. The sensitivity of the optimal allocation for the first region to changes in the visibility parameters of the second region is directly proportional to the current level of search effort within that region. One can, of course, directly determine the effects of these changes in the visibility parameters upon the optimal return. For example, increasing in the rate of the masking period causes an increase in the optimal return. Also, as one might expect, this effect diminishes to zero as the total available search time increases. On the other hand, the sensitivity of the optimal return to changes in the visibility rate increases with increasing available search time to an asymptotic value which is identical to that obtained for the previous model.

The situation in which the detection and visibility processes are discrete is modeled. The solution techniques utilized in the previous chapters were again applicable; however, the nature of the resulting detection functions

precluded the derivation of explicit expressions for the optimal allocation policies. Accordingly, numerical techniques were used. All of the extensions and special cases associated with the continuous detection models were shown to be valid in this situation.

6.1.4 Multiple Intervals of Visibility

In this model the First Allocation Rule (FAR) can differ from the classical FAR. Thus, the use of the latter in this situation could lead to erroneous selection of regions to be searched. The conditional detection functions range from concave to pseudo-concave depending upon the levels of Π, the prior probability on target visibility at the start of the search. For example, the assumption that the Π's are the limiting probabilities of an alternating renewal process leads to a concave conditional detection function.

The nature of the conditional detection functions for this model, conditioned on the target presence, preclude the derivation of explicit expressions for the optimal allocation policies. The pseudo-concave conditional detection functions give rise to optimal allocation policies which require minimum levels of search time before a new region is searched (as did the model of Chapter 4, which is, of course, a special case of this model). Approximate

solutions were derived from an analysis of the regional composite detection rates, which are functions of the visibility and detection parameters.

From these studies it is easy to determine the conditions under which the approximate allocation policy (which is accurate for large quantities of search effort) will be close to the classical allocation policies, as well as those for which the opposite conclusion holds. Thus, the classical or Koopman policy was shown (both numerically and analytically) to be adequate whenever (a) the search scenario is characterized by identical regional detection rates and identical regional visibility parameters, or (b) whenever the mean time until detection is much less than the mean length of the visibility periods.[1]

By choosing Π to be either zero or one, direct generalizations of the models of Chapters 2 and 3 are obtained. For example, for Π equal to unity, one obtains a multiple-interval-start-at-time-zero model, while the opposite situation yields a multiple-interval, random-start-time model.

Sensitivity analyses indicate that the sensitivity of the optimal allocation in a given region to changes in

[1]These results hold under the assumption that the Π's are the limiting probabilities of an associated alternating renewal process.

the visibility parameters is proportional to the amount of effort allocated to the region containing the visibility parameter, the constants of proportionality are given by the rates of change of the composite process rate with respect to the visibility parameters. These constants can be compared readily to determine the relative influence of the various visibility parameters.

It was also shown that the probability of detection tends toward unity with increasing search effort. The other models did not have this property, with the exception of the late-arrival, no departure model of Chapter 4. In light of these results, one may study the expected time problem without introducing the conditioning devise required for the other models. Numerical results indicate that, while the classical policy is adequate in many cases, for the problem of maximizing the probability of detection, it may not be efficient when minimizing the expected time until detection.

Finally, some general observations on our models are as follows. It was observed that (with a single exception), under the restrictions of

(a) a uniform distribution on target location,

(b) uniform regional detection capabilities,

(c) uniform visibility parameters, and

(d) levels of search time equal to twice the KET;

the Koopman or classical policies are approximately optimal in maximizing the probability of detection under a constraint on the total available search time. The exception occurs with the model of Chapter 4. Thus, the classical policies are applicable to a number of situations in which their applicability was not apparent. On the other hand, the inadequacy of the classical policies was demonstrated when one or more of the above restrictions is violated.

For each of the situations analyzed, it was assumed that the visibility and detection processes had very specific forms. The insights developed and summarized above were based on the use of these forms. However, the model structures are not restricted to these special forms, and accordingly, the optimal policies and returns can be readily obtained, via numerical techniques (e.g., dynamic programming), for any detection and visibility processes.

6.2 Areas of Future Research

In this section, several areas for future research are discussed. We briefly introduce each area and suggest some avenues of approach.

(1) Enrichments of the multiple interval model

We shall consider three directions of extension for this model:

(1) the target has a detection capability, i.e., if the target detects the searcher first, he exits;

(2) the number of transitions to the masked or visible state is controlled by an external random process, and

(3) the introduction of various levels of visibility.

We shall explore only the first direction.

One could consider two cases:

(1) the target can detect only while it is in the "visible" state,

(2) the target can detect the searcher in either state.

This situation gives a complete spectrum of visibility models, since, loosely speaking, it is equivalent to allowing a finite number of visibility intervals. The model of Chapter 5 allows for an infinite number and the models of Chapters 3 and 4 treat the single interval case. Note that the ultimate probability of detection in these situations is less than unity.

Let the visibility process of the target (within a region) be modeled as an alternating renewal process, the steady state probabilities of which give the probabilities

of the target being visible. First, we assume that the target's detection gear is working only while in the visible state (e.g., passive air search for a submarine). If the searcher enters the region and finds the target in the visible state, then the length of time the target spends in the visible state is a random variable given by the smaller of either:

(a) the forward recurrence time of the visibility period, or

(b) length of time required for the target to detect the searcher.

It might be assumed that the searcher has another chance at the target in case (a), while in case (b) the target has escaped.

Let the transition matrix for the model be given by

$$Q(T) = \begin{array}{c} \begin{array}{cccc} (V) & (I) & (D) & (E) \end{array} \\ \begin{pmatrix} 0 & R(t) & D(t) & H(t) \\ F_m(t) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array} ,$$

where the states are

V, visible;

I, masked;

D, target detected; and

E, target escaped.

Expressions for the distribution functions may be derived as in Chapter 5. Having obtained the transition matrix, one need only apply the procedures of Chapter 5 to obtain the conditional detection functions.

(2) Avalanche search model

In this model the searcher knows that a climber is lost in an avalanche or a storm somewhere on a mountain. The length of time that he may be expected to survive is a random variable dependent upon his precise location. The objective is to assign a limited amount of search effort in such a way as to maximize the probability that he will be found alive.

There are several other analogies of this model, the most obvious being the stricken submarine, e.g., the *Thresher* incident. Another interesting analogue is in the area of medical screening for a terminal disease. One is interested in the sequence, and quantity of treatment which maximizes the probability of survival.

A description of this problem may be obtained by starting with the general formulation of the model for a single interval of visibility starting at time zero, i.e., for the j^{th} region,

$$P(T_1) = p_j \int_0^{T_1} \rho_j(\tau)\overline{H}_j(\tau)d\tau \quad , \qquad (45)$$

where $\rho(\tau)$ is density function for the time until detection, $\bar{H}(\tau)$ is complementary cumulative distribution function on the length of the visible period, and p_j is the probability that the target is in the j^{th} region. Equation 45 is appropriate if the j^{th} region is searched first. Next, assume it has been determined that region K is next in sequence. It shall also be assumed that in going from region j to K, one incurs a switching time t_{jK}. The probability of detection is this region, assuming one allocates T_2 units of time, is given by

$$P_D(T_2) = p_K \int_0^{T_2} \rho_K(\tau)\overline{H}_K(\tau+T_1+t_{jK})d\tau \ , \qquad (46)$$

where p_K reflects the fact that the previous search of region j was unsuccessful. Conceptually, this process could be continued for successive numbers of regions. However, even if the sequence is given, one is still faced with the problem of determining the optimal quantities of the total available effort to assign to each region in the given sequence. This problem is more difficult than the previous formulations because of the non-separability aspects. If one were able to explicitly solve for the optimal allocations of a given sequence, then the substitution of

these results into the return function for the sequence would yield an expression which contained only the known search parameters. Having accomplished this, one could then test for dominance among all such return functions, hopefully, arriving at a criteria (in terms of the known parameters of the problem), which would enable the a priori determination of the optimal sequence.

This ambitious program was attempted for a 2-region search situation, with exponential distributions on the detection and visibility phenomena, with little success. The reason for the lack of success was the failure to obtain explicit expressions for the optimal allocations under a given sequence. However, this brief exercise did produce some interesting results and conjectures. In certain cases one places all the effort in the first region regardless of the total amount available, while in others, there is an upper bound on the amount of effort placed in the initial region. It was also noted that the First Allocation Rule will determine the proper order for very small quantities of total available search effort. Finally, the research suggests the following conjecture: "For the N-region problem with $p_i k_i = p_j k_j$, $j \neq i$, the optimal sequence is obtained from the ranking as to increasing expected lengths of the respective visibility periods."

(3) Random search times

One might consider either of the following random search time situations: First, the case in which the total available effort is a random variable with known distribution; second, the situation in which the searcher spends a random amount of time in each region. The former situation could arise in context of an aerial search in which unknown or changing weather conditions randomly influence the total search time. An approach to this situation might be to formulate it as a stochastic programming problem of either the 2-stage or chance-constraint variety depending upon the particular situation. One approach to the second problem might be to solve it, assuming the search times to be fixed. Then use these optimal values as nominal values with some known, possibly regionally dependent, variation to compute the expected probability of detection. Or as an approximation to this, one could compute the partial derivatives (evaluated at the nominal levels) of the return function with respect to the optimal allocations. These scenarios become especially interesting in view of the allocation policies for the models of Chapters 4 and 5, since models gave rise to situations in which the optimal policy has the property that the amount of effort put in a given region is not monotone non-decreasing as a function of the total available search effort (e.g., see Figure 4, Chapter 4).

(4) Target motion

The process of target transitions from one region to another might be realistically modeled as a semi-Markov process with the distribution of transition times depending upon the current state and the next state. Such a model seems to be appropriate in the sense that a commander can choose both the next region and the transit speed to that region at random thus generating the semi-Markov transition matrix. In actual transit situations this appears to be a more reasonable model than, say, Brownian motion. In addition to the motion consideration, target intervisibility could occur due to changes in velocity. In this same context, tracking states (i.e., entrances to and exits from) are also of interest. With the introduction of such states, some new search objective functions might be of interest, e.g., the maximization of the probability that the time spent in the tracked state is greater than some specified level.

Appendix A
METHODOLOGY

A.1 Purpose

The intent of this appendix is to develop in detail the solution methodology for the two major search objectives:

(a) maximizing the probability of detection under a constraint on the available effort, and

(b) minimizing the expected time until detection.

First, we shall present a modified version of the Charnes-Cooper Algorithm for the discrete search space version of Koopman's problem. This algorithm is used in solving the approximations to many of the models presented in Chapters 2-5. The discrete detection version of the Koopman problem is then discussed and related to the continuous detection analog. The technique presented will be useful in analyzing the discrete versions of many of the visibility models.[1] Finally, we look at minimizing the expected time until detection for both the continuous and discrete cases.

A.2 The Allocation of Search Effort to Maximize the Probability of Detection

A.2.1 Continuous Detectors

In this section we modify the Charnes-Cooper (1958) Algorithm for solving the discrete search space problem.

[1] See Appendix B for discrete detector formulations and results.

The target of Koopman's problem is located in one of N regions with prior probability vector $P = (p_1, p_2, \ldots, p_N)$ and the searcher has detection rates for each of these regions given by the vector $K = (k_1, k_2, \ldots, k_n)$. The basic optimization problem becomes

$$\max \sum_{i=1}^{N} p_i \; (1 - e^{-k_i t_i}) \; ,$$

$$\text{S.T.} \sum_{i=1}^{N} t_i = T \; ,$$
$$t_i \geq 0 \; .$$

The Charnes and Cooper algorithm for the above problem has the restriction that the $k_i = k$ for each region, which can be easily removed. The application of the Kuhn-Tucker conditions to the above problem will yield the optimal solution. Then the conditions are necessary and sufficient in this problem since we have a concave objective function. Reformulating the problem as a minimization one:

$$\max \sum_{i=1}^{N} p_i(1 - e^{-k_i t_i}) \equiv \min \sum_{i=1}^{N} p_i e^{-k_i t_i}$$

Scaling the allocations by T;

$$\frac{t_i}{T} = \tau_i \text{ , and } \sum_{i=1}^{N} \tau_i = 1,$$

one obtains as the problem

$$\min \sum_{i=1}^{N} p_i e^{-Tk_i \tau_i}$$

$$\text{S.T.} \quad \sum_{i=1}^{N} \tau_i = 1$$

$$\tau_i \geq 0 \; .$$

Form the Lagrangian

$$W(\tau,w,\psi) = \sum_{j=1}^{N} p_j e^{-Tk_j \tau_j} - \sum_{j=1}^{N} w_j \tau_j + \psi \sum_{j=1}^{N} (\tau_j - 1) \; .$$

Application of the Kuhn-Tucker conditions to the above equation yields

$$Tk_j p_j e^{-Tk_j \tau_j} = \psi - w_j \quad \text{if } w_j \qquad \tau_j \geq 0 \; ,$$

$$Tk_j p_j e^{-Tk_j \tau_j} = \psi, \text{ i.e., for } \tau_j > 0, \text{ then } w_j = 0 \; .$$

From these equations one obtains that

$$Tk_j p_j = \psi - w_j \text{ when } \tau = 0 \; , \quad w_j \geq 0 \; .$$

$$Tk_j p_j e^{-Tk_j \tau_j} = \psi \quad \text{when } \tau_j > 0 \; .$$

Taking logarithms, we have

$$\ln T + \ln k_j + \ln p_j \leq \ln\psi \tag{1}$$

$$\ln T + \ln k_j - Tk_j\tau_j + \ln p_j = \ln\psi, \quad \tau_j > 0 \ . \tag{2}$$

Dividing (2) by k_j, one obtains

$$\frac{\ln T}{k_j} + \frac{\ln k_j}{k_j} - T\tau_j + \frac{\ln p_j}{k_j} = \frac{\ln\psi}{k_j}, \quad \tau_j > 0 \ . \tag{3}$$

Let the set J be the set of indices of those regions which receive a positive allocation of search effort, i.e., note that

$$J = \left\{ i \,\middle|\, \tau_i > 0 \right\} ,$$

and

$$\sum_{j=1}^{N} \tau_i = \sum_{j \epsilon J} \tau_j = 1 \ .$$

Summing equation 3 over all $j \ \epsilon \ J$,

$$\sum_{j\epsilon J} \frac{\ln T}{k_j} + \sum_{j\epsilon J} \frac{\ln k_j}{k_j} - T\sum_{j\epsilon J} \tau_j + \sum_{j\epsilon J} \frac{\ln p_j}{k_j} = \sum_{j\epsilon J} \frac{\ln\psi}{k_j}$$

or

$$\ln\psi = \ln T + \frac{1}{\sum_{j\epsilon J} \frac{1}{k_j}} \left(\sum_{j\epsilon J} \frac{\ln k_j}{k_j} + \frac{\ln p_j}{k_j} - T \right) \tag{4}$$

Denote lnx by $\hat{x}$ and note that $Tk_j\tau_j > 0$ for $j \varepsilon J$. Then, using (1) and (4)

$$\hat{T} + \hat{k}_S + \hat{p}_S \leq \hat{T} + \frac{1}{\sum\limits_{j\varepsilon J} \frac{1}{k_j}} \left(\sum_{j\varepsilon J} \frac{\hat{k}_j}{k_j} + \frac{\hat{p}_j}{k_j} - T \right)$$

$$< \hat{T} + \hat{k}_r + \hat{p}_r \tag{5}$$

where $S \not\varepsilon J$ and $r \varepsilon J$

Substitution of (4) into (2) yields

$$\tau_j = \frac{1}{Tk_j}\left[\hat{p}_j + \hat{k}_j - \frac{1}{\sum\limits_{i\varepsilon J} \frac{1}{k_j}} \left(\sum_{j\varepsilon J} \frac{\hat{k}_j}{k_j} + \frac{\hat{p}_j}{k_j} - T \right)\right]. \tag{6}$$

Now, if the members of the set J were known, the solutions are given by (6). Toward this end we modify a lemma by Charnes and Cooper.

From (5) we have

$$\min_{r\varepsilon J} \hat{p}_r + \hat{k}_r > \frac{1}{\sum\limits_{j\varepsilon J} \frac{1}{k_j}} \left(\sum_{j\varepsilon J} \frac{\hat{p}_j}{k_j} + \frac{\hat{k}_j}{k_j} - T \right) \tag{7}$$

$$> \max_{S\varepsilon J} \hat{p}_S + \hat{k}_S$$

Replace $\hat{p}_j + \hat{k}_j$ by β_j. The resultant lemma is

Lemma: If $\beta_{m+1} > \frac{1}{\sum_{i=1}^{m} \frac{1}{k_j}} \left(\sum_{i=1}^{m} \frac{\beta_i}{k_i} - T \right)$,

then

$$\beta_{m+1} > \frac{1}{\sum_{i=1}^{m+1} \frac{1}{k_i}} \left(\sum_{i=1}^{m+1} \frac{\beta_i}{k_i} - T \right).$$

Proof: By hypothesis

$$\sum_{i=1}^{m} \frac{1}{k_i} \beta_{m+1} > \sum_{i=1}^{m} \frac{\beta_i}{k_i} - T$$

$$\sum_{i=1}^{m} \frac{1}{k_i} \beta_{m+1} + \frac{\beta_{m+1}}{k_{m+1}} > \sum_{i=1}^{m} \frac{\beta_i}{k_i} + \frac{\beta_{m+1}}{k_{m+1}} - T$$

$$\beta_{m+1} > \frac{1}{\sum_{i=1}^{m+1} \frac{1}{k_{m+1}}} \left(\sum_{i=1}^{m+1} \frac{\beta_i}{k_i} - T \right)$$

Q.E.D.

To apply these results, to our minimization problem

(a) Arrange the values of $\beta_j = \hat{p}_j + \hat{k}_j$ in decreasing order, and relabel regions according to this order.

(b) If $\hat{p}_1 + \hat{k}_1 > k_1 \left(\frac{\hat{p}_1 + \hat{k}_1}{k_1} - T \right)$, region 1 is selected.

(c) If $\hat{p}_2 + \hat{k}_2 \leq k_1 \left(\frac{\hat{p}_1 + \hat{k}_1}{k_1} - T \right)$, stop.

Otherwise region 2 is selected and one continues via the lemma until the set J is determined.

The results for the two-region model follow directly. We shall assume that $p_1k_1 \geq p_2k_2$. First, we determine under what conditions all the search effort is placed in region 1. From the algorithm for the selection of indices for the set J, one stops at region 1 if

$$\ln(p_2k_2) \leq k_1\left(\frac{\ln(p_1k_1)}{k_1} - T\right)$$

leading to

$$T^* \leq \frac{1}{k_1} \ln\left(\frac{p_1k_1}{p_2k_2}\right) . \tag{8}$$

Then for total search time $T \leq T^*$, one spends the entire time searching region 1 . For $T \geq T^*$, one obtains, using (6), the optimal allocations to the two regions as

$$t_1 = \frac{1}{k_1 + k_2}\left[\ln\left(\frac{p_1k_1}{p_2k_2}\right) + k_2T\right] ,$$

and

$$t_2 = \frac{1}{k_1 + k_2}\left[\ln\left(\frac{p_2k_2}{p_1k_1}\right) + k_1T\right] . \tag{9}$$

We observe that for large T, one has

$$t_1 \approx \frac{k_2T}{k_1 + k_2} , \quad t_2 \approx \frac{k_1T}{k_1 + k_2}$$

The optimal value for the probability of detection is then given by

$$P(T) = p_1\left[1 - \left(\frac{p_1k_1}{p_2k_2}\right)^{-\frac{k_1}{k_1+k_2}} e^{-\frac{k_1k_2}{k_1+k_2}T}\right]$$

$$+ p_2\left[1 - \left(\frac{p_2k_2}{p_1k_1}\right)^{-\frac{k_2}{k_1+k_2}} e^{-\frac{k_1k_2}{k_1+k_2}T}\right].$$

To handle the N-region case, one merely follows the procedure outlined in the proof of the algorithm

A.2.2 Discrete Detectors

Recall the following definition given in Chapter 1, Incremental Detection: Let $1 - q$ be the instantaneous probability of detection on any scan of a region. Given m such scans, the conditional probability of detection is

$$P(m) = 1 - q^m .$$

Let $P = (p_1, p_2, \ldots, p_N)$ be the prior probabilities on target location. Then the N-region search problem may be stated as

$$\max \sum_{i=1}^{N} p_i(1 - q_i^{m_i})$$

$$\text{S.T.} \quad \sum_{i=1}^{N} m_i \leq M \qquad (10)$$

$$m_i \geq 0 \text{ , integers.}$$

This formulation may be solved using Dynamic Programming, (or the results of Chew (1967)), however, some results (Wagner, 1969) enable one to take a more direct approach. These results are summarized in the following theorem.

Wagner's Theorem

Let e and c be real-valued functions (of two variables) defined on

$\{(x,j) \mid a < x < b,\ \ell(x) \leq j \leq u(x),\ j \text{ an integer}\}$.

Let Φ be the set of all integer-valued functions f on (a,b) such that $\ell(x) \leq f(x) \leq u(x)$ for $a < x < b$ for which

$$-\infty < E(f) = \int_a^b e(x,f(x))dx < \infty$$

$$-\infty < C(f) = \int_a^b c(x,f(x))dx < \infty .$$

Suppose that $g \in \Phi$ has the following property: there exists a $\lambda > 0$ such that for all $x \in (a,b)$ and integers j

$$e(x,j) - e(x,j-1) \leq \lambda[c(x,j) - c(x,j-1)]$$

whenever $\overline{g(x)} < j \leq \overline{u(x)}$, and

$$e(x,j) - e(x,j-1) \geq \lambda[c(x,j) - c(x,j-1)]$$

whenever $\ell(x) \leq j - 1 < g(x)$, then

$$E(g) = \max \{E(f) | f \in \phi \text{ and } C(f) \leq C(g)\}$$

$$C(g) = \min \{C(f) | f \in \phi \text{ and } E(f) \geq E(g)\} .$$

Application of the theorem to (10) gives the following necessary conditions on the solution:

(a) for the i^{th} region

$$m_i = \left[\frac{\log \left(\frac{\lambda}{p_i(1-q_i)} \right)}{\log q_i} + 1 \right] , \qquad (11)$$

where the bracket implies the largest integer therein,

(b) The objective is to choose a $\lambda > 0$, such that (a) is satisfied for all i for which $m_i > 0$, and

$$\sum_{i=1}^{N} m_i = M .$$

One does this iteratively by assuming a λ, checking (b), etc.

In the two-region situation, one has for the optimal allocations

$$m_1 = \frac{1}{\log q_1 + \log q_2}\left[(M-2)\log q_2 + \log\left(\frac{p_2(1-q_2)}{p_1(1-q_1)}\right)\right] + 1\ ,$$
$$m_2 = \frac{1}{\log q_1 + \log q_2}\left[(M-2)\log q_1 + \log\left(\frac{p_1(1-q_1)}{p_2(1-q_2)}\right)\right] + 1\ . \qquad (12)$$

Next, we shall use these results to answer some questions about the limiting values of the m_i. In the limit as $M \to \infty$, from (11) one has that

$$m_i \approx \frac{\log \lambda}{\log q_i}\ . \qquad (13)$$

For the two-region situation, this results in the following allocations

$$m_1 = \frac{(\log q_2)\ M}{\log q_1 + \log q_2}$$

and (14)

$$m_2 = \frac{(\log q_1)\ M}{\log q_1 + \log q_2}$$

If one assumes

(a) $k_i \Delta t = 1 - q_i$

(b) $q_i \to 1^-$,

then the optimal allocations for the discrete and continuous detectors are approximately equal. This agrees with Pollock's (1960) observations on the discrete detection problem. In general, one might make good use of the continuous analog in order to obtain an initial solution for the discrete necessary conditions, even in the N-region case.

A.3 Minimization of the Expected Time Until Detection

A.3.1 Continuous Detectors

Here we consider the other classic objective function, the minimization of the expected time until detection with no constraint on the available search time. Dobbie (1963) has shown that a concave conditional detection function is sufficient to guarantee that the policy which maximizes the probability of detection for each value of T, the available search time, is also the policy which minimizes the expected time until detection. Certainly, the conditional detection function obtained from the "law of random search" satisfies the above condition. We shall use this result to obtain an expression for the minimum expected time until detection.

The optimal value of the probability of detection for a two-region search situation, given T, is

$$P(T) = \sum_{i=1}^{2} p_i(1 - e^{-k_i t_i(T)}) ,$$

and

$$\frac{dP(T)}{dT} = \sum_{i=1}^{2} p_i k_i e^{-k_i t_i(T)} \frac{dt_i(T)}{dT}$$

where, from the previous section, the $t_i(T)$ are given by (assuming region 1 is selected first)

$$t_1(T) = \begin{cases} T & , \quad T < T^* \\ \frac{1}{k_1 + k_2}\left[\ln\left(\frac{p_1 k_1}{p_2 k_2}\right) + k_2 T\right] , & \quad T \geq T^* \end{cases}$$

$$t_2(T) = \begin{cases} 0 & , \quad T < T^* \\ \frac{1}{k_1 + k_2}\left[\ln\left(\frac{p_2 k_2}{p_1 k_1}\right) + k_1 T\right] , & \quad T \geq T^* \end{cases} .$$

The minimal expected search time will then be given by

$$E \equiv \int_0^\infty (1 - P(T))dT .$$

Carrying out the above integration, one obtains for the minimal expected time to detect

$$E = p_1 \left\{ \frac{1}{k_i} \left(1 - e^{-k_i T^*} \right) + \left(\frac{p_1 k_1}{p_2 k_2} \right)^{-\frac{k_1}{k_1 + k_2}} \right. \tag{14}$$

$$\left. \cdot \left(1 + \frac{k_1}{k_2} \right) \left(\frac{k_1 + k_2}{k_1 k_2} \right) e^{-\frac{k_1 k_2}{k_1 + k_2} T^*} \right\} + p_2 T^* ,$$

where T^* is given by (8).

For the situation in which the infinitesimal detection probabilities are identical, i.e., $p_1 k_1 = p_2 k_2$, the above expression simplifies to

$$E = \frac{k_1 + k_2}{k_1 k_2} .$$

A.3.2 Discrete Detectors

Here we need only recall the important results of Black (1965). He shows that the policy with the minimum expected cost is generated by the rule" "Always look in the region for which the posterior probability (given the failure of earlier looks) of finding the object divided by the cost (time) is maximum."

A.4 First Allocation Rule

This section considers the following question: For the N-region, discrete search space situation, how does one determine the order in which the regions receive search effort as the amount of total available search time increases? As the amount of available search time is

increased one can begin to place the additional effort in regions which have yet to receive any effort. In the previous section, it was noted that this rule, which we shall designate as the *First Allocation Rule* was determined by choosing as the initial region that region i for which

$$p_i k_i = \max_{1 \le j \le N} p_j k_j \ .$$

The First Allocation Rule for the general case

$$\max \sum_{i=1}^{N} p_i P_i(t_i)$$

$$\text{S.T.} \sum_{i=1}^{N} t_i \le T \ ,$$

$$t_i \ge 0 \ ,$$

can be established as follows. First, we observe that if $T = \varepsilon << 1$, then it is important to allocate T to that region for which the rate of return is maximized,

$$P_i'(T) = \max_{1 \le j \le N} P_j'(T) \ .$$

Thus we can determine the initial region. Suppose next that one has T sufficiently large to warrant the search of an additional region. Once again the goal is the maximization of the return from that additional effort, hence one chooses

$$P_i'(\varepsilon) = \max_{k \in J} P_k'(\varepsilon) ,$$

where $J = \{i | t_i > 0\}$. If the conditional detection function has the property that $P'(0) > 0$, then one can let $\varepsilon \to 0$ in the above rules; if $P'(0) = 0$, then one merely chooses $\varepsilon \ll 1$ and applies the rules as stated. We remark that this technique does not enable one to determine the value of T at which the next region begins to receive effort, but merely the ordering of the regions.

Appendix B

DISCRETE DETECTION AND VISIBILITY PROCESSES

B.1 Introduction

This appendix contains the formulations and results for the discrete analogues of the models of Chapters 2-4 respectively. Explicit expressions for the optimal allocation policies are developed for each of the models.

B.2 The Binary Model

If the searcher is utilizing a discrete (glimpse) detection device, two versions of the Binary model may be considered. First assume that upon the entry of the searcher the target holds his current visibility status throughout the search. In this situation, the problem of maximizing the probability of detection is given by

$$\max \sum_{i=1}^{N} p_i v_i (1 - q_i^{m_i}) \quad ,$$

$$\text{S.T.} \sum_{i=1}^{N} m_i \leq M \quad , \tag{1}$$

where M and m_i are positive integers and $1 - q_i$ is the glimpse detection probability in the i^{th} region. The second situation occurs under the assumption that the target's visibility status changes with each glimpse. The problem

of maximizing the probability of detection is stated as

$$\max \sum_{i=1}^{N} p_i \left\{1 - (q_i v_i)^{m_i}\right\}$$

$$\text{S.T.} \sum_{i=1}^{N} m_i \leq M, \; m_i \geq 0. \tag{2}$$

These discrete detection models can be solved in a variety of ways. In order to directly compare the discrete and continuous detector models, we shall present the solution given in Appendix A. First, under the assumptions leading to (1), the optimal allocations (for the two-region problem) are

$$m_1 = \frac{1}{\ln q_1 + \ln q_2} \left\{\ln q_2 (M-2) + \ln \left(\frac{p_2 v_2 (1-q_2)}{p_1 v_1 (1-q_1)}\right)\right\} + 1,$$

and

$$m_2 = \frac{1}{\ln q_1 + \ln q_2} \left\{\ln q_1 (M-2) + \ln \left(\frac{p_1 v_1 (1-q_1)}{p_2 v_2 (1-q_2)}\right)\right\} + 1.$$

As $M \to \infty$, one has

$$m_1 = \frac{(\ln q_2) M}{\ln q_1 + \ln q_2}$$

and

$$m_2 = \frac{(\ln q_1) M}{\ln q_1 + \ln q_2} \quad . \tag{3}$$

Under the assumptions leading to (2), the analogous policies are

$$m_1 = \frac{1}{\ln v_2q_2 + \ln v_1q_1}\left\{(M-2)\ \ln(v_2q_2) + \ln \frac{p_2(1-v_2q_2)}{p_1(1-v_2q_2)}\right\} + 1$$

and

$$m_2 = M - m_1.$$

As $M \to \infty$, we have[1]

$$m_1 = M \frac{\ln v_2q_2}{\ln v_2q_2 + \ln v_1q_1} \quad ,$$

and (4)

$$m_2 = M \frac{\ln v_1q_1}{\ln v_1q_1 + \ln v_2q_2} \quad .$$

If one assumes that $k_i\Delta t \simeq 1 - q_i$, where Δt is the "glimpse" time, then as $q_i \to 1$, one has from (3) that the optimal allocations for the continuous and discrete cases are approximately equal.

Finally, note that the results of Chew (1967) given in Chapter 1 are directly applicable here. Namely, "to maximize the probability of finding the object in a fixed number of searches, choose those n-searches for which the posterior probability (given the failure of earlier looks) of finding the object is largest." We calcualte these probabilities using the operator θ, i.e., if $\mathbb{P} = (p_1,\ldots,p_N)$ then

---

[1] In each of the above expressions for m_i, one must choose the largest integer in the right hand side of the appropriate expression.

$(\theta_i \mathbb{P})_j$, is the posterior probability that the target is in the j^{th} region given the failure of the past search in region i. For the model given by (1),

$$(\theta_i \mathbb{P})_j = \begin{cases} \dfrac{q_i v_i p_i}{1 - (1 - q_i) p_i v_i}, & j = i \\ \\ \dfrac{p_j}{1 - (1 - q_i) p_i v_i}, & j \neq i \end{cases} . \tag{5}$$

The model given by (2) yields the following expressions

$$(\theta_i \mathbb{P})_j = \begin{cases} \dfrac{(1 - (1 - q_i) v_i) p_i}{1 - (1 - q_i) v_i p_i}, & j = i \\ \\ \dfrac{p_i}{1 - (1 - q_i) v_i p_i}, & j \neq i \end{cases} . \tag{6}$$

Clearly, as $v_i \to 1$ for all i, both relate to the model first studied by Pollock (1960).

B.3 A Random Interval Process: Target Initially Visible

Let $0 < h_j < 1$ denote the probability that the target is visible on a glimpse in the j^{th} region. It is assumed that the target is visible at the start of the search. Given that the target is visible at the start of the search, and that the target is present in the j^{th} region, the conditional probability of detecting it, at or before the m^{th} look, is given by

$$P_j(m) = h_j\bar{q}_j + \bar{q}_j h_j^2 q_j + \cdots + \bar{q}_j h_j^m q_j^{m-1}$$

where $\bar{q}_j = (1 - q_j)$ = the glimpse probability of detection in the j^{th} region, and

$$P_j(m) = \frac{h_j\bar{q}_j}{1 - h_j q_j}(1 - (h_j q_j)^m) .$$

Of course as $h \to 1$, this yields Pollock's (1960) model. The problem of maximizing the probability of detection, given a total of M glimpses, is

$$\text{MAX} \sum_{i=1}^{N} p_i\left(\frac{h_i\bar{q}_i}{1 - q_i h_i}\right)(1 - (h_i q_i)^{m_i})$$

$$\sum_{i=1}^{N} m_i \leq M ,$$

where m_i are non-negative integers.

Again the results of Appendix A are applicable. We replace

$$p_i \text{ by } \frac{p_i h_i \bar{q}_i}{1 - q_i h_i} \text{, and } q_i \text{ by } h_i q_i$$

in equation 12, Section A.2 to obtain (in the two-region case) the optimal allocations

$$m_1 = \left[\frac{1}{\ln (h_1 q_1) + \ln (h_2 q_2)} \left\{ [M - 2] \ln (h_2 q_2) + \left(\frac{p_1 h_1 (1 - q_1)}{p_2 h_2 (1 - q_2)} \right) \right\} + 1 \right] , \tag{8}$$

and

$$m_2 = [M - m_1] .$$

As $M \to \infty$, (8) yields

$$m_1 \approx \frac{M \ln (h_2 q_2)}{\ln (h_1 q_1) + \ln (h_2 q_2)} ,$$

and (9)

$$m_2 \approx \frac{N \ln (h_1 q_1)}{\ln (h_1 q_1) + \ln (h_2 q_2)} .$$

Setting up the correspondence $k\Delta t = 1 - q$ and $\lambda\Delta t = 1 - h$, where Δt is the "glimpse" time, one can obtain the corresponding allocation policy for continuous detectors under the assumption that k and $\lambda < < 1$. The continuous policy may be useful in obtaining initial estimates for the discrete policy via the above correspondence.

B.4 Random Interval Process: Random Initiation and Limit

Making use of the discrete model just obtained, we observe, given that the target becomes present at the j^{th} glimpse, that the probability of detection in $m - j + 1$ identical glimpses is given by

$$P(m|j) = h\bar{q} + h^2\bar{q}q + \cdots + \bar{q}h^{m-j+1}q^{m-j}$$

$$= \frac{h\bar{q}(1 - (hq)^{m-j+1}}{1 - hq} \quad ,$$

where

$\bar{q} = 1 - q$ = glimpse probability of detection,

h = probability target remains visible.

Let A denote the probability that the target appears on any glimpse, then the probability that the target becomes visible on the i^{th} glimpse is $P(i) = A(1 - A)^{i-1}$. The probability of detection in m glimpses is

$$P(m) = \sum_{i=1}^{m} P(m|i)P(i) = \sum_{i=1}^{m} \frac{h\bar{q}}{(1 - hq)} (1 - (hq)^{m-j+1})A(1 - A)^{i-1} \tag{10}$$

If $A = 1$, $P(m) = \frac{h\bar{q}}{1 - hq}(1 - (hq)^m)$, which is the previous model. Carrying out the summation in (10)

$$P(m) = \frac{sA\bar{q}}{(1 - sq)}\left[\frac{1 - (1 - A)^m}{A} - sq\,\frac{(sq)^m - (1 - A)^m)}{(sq - (1 - A))}\right].$$

The discrete, N-region optimization problem is

$$\max \sum_{i=1}^{N} P_i \,\frac{h_i A_i \bar{q}_i}{1 - h_i q_i}\left[\frac{1 - (1 - A_i)^{m_i}}{A_i} - \frac{h_i q_i (h_i q_i)^{m_i} - (1 - A_i)^{m_i}}{(h_i q_i - (1 - A_i))}\right] \tag{11}$$

$$\text{S.T.} \sum_{i=1}^{N} m_i \le M$$

$$m_i \ge 0,\ \text{integers}.$$

We make the following observations concerning P(m):

(a) $\lim_{A \to 1^-} P(m) = \frac{h\bar{q}}{1 - hq}\left\{(1 - hq)^m\right\}$, the model of B.2

(b) $\lim_{m \to \infty} P(m) = \frac{h\bar{q}}{1 - hq}$, the limiting result for the model of B.2.

(c) $\lim_{\substack{h \to 1^- \\ A \to 1^-}} P(m) = \left\{1 - q^m\right\}$, standard discrete detector model of Pollock (1960)

(d) $\lim_{h \to 1^-} P(m) = \left\{1 - \frac{Aq^{m+1}}{(q - (1 - A))} + (1 - A)^m\left\{\frac{Aq}{q - (1 - A)} - 1\right\}\right\}$

a late arrival, no departure model.

The results of Wagner, given in Appendix A are again applicable to equation 11; however, just as in the continuous analog of this model, the solution techniques are considerably more difficult. The following conditions lead to limiting allocation policies similar to those already obtained:

(a) $M \gg 1$, and $A_i \to 1^- \ \forall_i$, then the limiting allocations for two regions are

$$m_1 \approx \left[\frac{M \ln[h_2 q_2]}{\ln[h_2 q_2] + \ln\ [h_1 q_1]}\right]$$

and (12)

$$m_2 \approx \left[\frac{M \ln[h_1 q_1]}{\ln[h_2 q_2] + \ln[h_1 q_1]}\right].$$

(b) $M \gg 1$, and $A_i \ll 1$, $_i$, we obtain for the two-region case[1]

$$m_1 \approx \left[\frac{M \ln[1 - A_2]}{\ln[1 - A_2] + \ln[1 - A_1]}\right]$$

and (13)

$$m_2 \approx \left[\frac{M \ln[1 - A_1]}{\ln[1 - A_1] + \ln[1 - A_1]}\right]$$

which simplifies to

$$m_1 \approx \frac{A_2 M}{A_1 + A_2} \text{ and } m_2 \approx \frac{A_1 M}{A_1 + A_2} \ .$$

To set up the correspondence to the continuous model, let

[1][ ] denotes the largest integer therein.

$$\lambda \Delta t \sim (1 - s)$$
$$k \Delta t \sim (1 - q)$$
$$\beta \Delta t \sim (A) ,$$

where Δt is the "glimpse" time, then as $h \to 1^-$, $q \to 1^-$, and $A \to 0^+$, the continuous detection policy approximates the discrete policy.

Appendix C

CONTINUOUS SEARCH SPACE FORMULATIONS

C.1 Introduction

In this appendix we present the continuous search space formulations for the models of Chapters 2-4. We restrict our attention to continuous detection models throughout our discussions. The models are derived in the next sections in the order in which they are introduced in the text. Solution techniques are investigated in the final section.

C.2 Model Development

In order to present the continuous search space version of the Binary visibility model, we introduce the following notation. Let

- $p(x)$ be the probability density function on target location, defined on a region A;
- $v(x)dx$ denote the probability that a target in $(x, X + dx)$ is visible; and
- $\phi(x)$ denote the search intensity function, defined as the amount of search effort (time) allocated to an interval $(x, x + dx)$.

The conditional probability of detecting (conditioned on the target being visible and present) a target at x with effort of intensity $\phi(x)$ is denoted by $P(x, \phi(x))$. Using the above definitions, the problem of maximizing the probability of detection over a continuous search space is

$$\max \int_A v(x)p(x)P(x,\phi(x))dx \tag{1}$$

$$\text{S.T.} \int_A \phi(x)dx = T \quad ,$$

and

$$\phi(x) \geq 0.$$

The continuous search space version of the random interval (initially visible) model is

$$\max \int_A p(x)M(x,\phi(x))$$

$$\text{S.T.} \int_A \phi(x)dx \leq T \; , \tag{2}$$

$$\phi(x) \geq 0 \quad ,$$

where

$M(x,\phi(x)) = \int_0^{\phi(x)} \rho(x,\tau)\bar{H}(x,\tau)d\tau$ = conditional probability of detecting a target in the interval $(x,x + dx)$, conditioned on the target being present,

$\bar{H}(x,\tau)$ = probability that the length of the visibility period at x is greater than or equal to τ.

$\rho(x,\tau)d\tau$ = probability that the time until detection at x is in the interval $(\tau, \tau + d\tau)$, given that the target is continuously visible at x up to $\tau + d\tau$.

The random interval (random initiation and limit) model in the continuous search space situation is

$$\max \int_A p(x) \int_0^{\phi(x)} f(x,\mu)M(x,\phi(x) - \mu)\, d\mu dx \tag{3}$$

$$\text{S.T.} \int_A \phi(x)dx \le T$$

$$\phi(x) \ge 0 \ ,$$

where

$f(x,\mu)d\mu$ = probability that a target at x becomes visible in the interval $(\mu,\mu+d\mu)$.

$M(x,\phi(x)-\mu) = \int_0^{\phi(x)-u} \rho(x,\tau-\mu)\bar{H}(x,\tau-\mu)d\tau$, in the notation of the previous paragraph.

C.3 Solution Techniques

As noted in the literature review Zahl (1963) gave necessary and sufficient conditions for a general version of the search problem, i.e.,

$$\max F(\phi) = \int_c^d f(x,\phi(x))dx$$

$$\text{S.T. } G(\phi) = \int_c^d g(x,\phi(x))dx = T \ ,$$

and

$$a(x) \leq \phi(x) \leq b(x) \ .$$

His major result is the following theorem:

Theorem

A necessary and sufficient condition that a function $\bar{y}(\cdot)$ maximize $F(\phi(\cdot))$, under the above restrictions, is that there exist a constant λ such that for almost every x, $\bar{y}(x)$ maximizes

$$f(x,\phi) - \lambda g[x,\phi] \text{ over } a(x) \leq \phi(x) \leq b(x) \ .$$

The correspondence to our work is

$$f(x,\phi(x)) = p(x)M(x,\phi(x)) ,$$

where $p(x)$ is the probability density function on target location and $M(x,\phi(x))$ is the conditional detection function which results from one of our models. The function $g(x,\phi(x))$ becomes, in our case, just $\phi(x)$. Hence we are concerned with finding a constant λ and a function $\bar{y}(x)$ which maximizes

$$p(x)M(x,\phi(x)) - \lambda\phi(x) .$$

Thus we shall be interested in finding the solution to the functional equation

$$p(x)M_{\phi(x)}(x,\phi(x)) - \lambda = 0 . \qquad (4)$$

In general, one cannot expect to be able to solve (4) for $\phi(x)$ as a function of λ. However, in the situation in which $M(x,\phi(x))$ satisfies the conditions of deGuenin (1961), then the existence of an inverse function for $M_{\phi(x)}(x,\phi(x))$, is guaranteed;[1] i.e.,

$$\phi(x) = M^{-1}_{\phi(x)} \left(\frac{\lambda}{p(x)}\right) .$$

Now we proceed as follows

(a) Choose $\lambda_1 > 0$, which determines a subset $A \subset [c,d]$ such that

$$\phi(x) > 0 \text{ for } x \in A .$$

[1] See Chapter 1 for a complete description of these conditions

(b) then form

$$\int_A \phi(x)dx = T_1$$

(c) If

$T_1 = T$, $\phi(x)$ is optimal

$T_1 > T$, $\lambda_2 > \lambda_1$ and go to step a

$T_1 < T$, choose $\lambda_2 < \lambda_1$ and go to step a

We investigate the applicability of de Guenin's conditions to the models just developed. First, for the Binary model, one has the following optimization problem.

$$\max \int_R p(x)v(x)\left[\int_0^{\phi(x)} \rho(x,\tau)d\tau\right]dx$$

$$\text{S.T.} \int_R \phi(x)dx = T \ ,$$

$$\phi(x) \geq 0 \ .$$

Since $v(x) \geq 0$ for all x, if $P(x, \phi(x)) = \int_0^{\phi(x)} \rho(x,\tau)d\tau$ satisfies de Guenin's conditions, then certainly $v(x)P(x,\phi(x))$ will also. For the model of Chapter 3, the problem statement is given by

$$\max \int_R p(x)\left[\int_0^{\phi(x)} \alpha(x,\tau)\bar{H}(x,\tau)d\tau\right]dx \ .$$

$$\text{S.T.} \int_R \phi(x)dx = T$$

$$\phi(x) \geq 0 \, .$$

Now conditions (1), (2), and (4) (See Chapter 1) certainly hold; if one assumes that $P(x,\phi(x))$ satisfies the conditions of de Guenin, condition (3) follows. For the model of Chapter 4, we have

$$\max P(x,\phi(x)) = \int_R p(x)\left[\int_0^{\phi} f(x,\mu) \cdot \left[\int_0^{\phi-\mu} \rho(x,\tau)\bar{H}(x,\tau)d\tau\right]d\mu\right]dx$$

$$\text{S.T.} \int_R \phi(x)dx = T$$

$$\phi(x) \geq 0 \, .$$

Here an analysis of the second partial derivative of $P(x,\phi(x))$ with respect to $\phi(x)$ yields the result that $dP(x,\phi(x))/d\phi(x)$ is *not* a decreasing function of ϕ. Hence the required inverse function does not exist. For the models of Chapter 5, one cannot make a general statement; however, several of the examples therein violate Condition (3) under certain conditions.

In the event the conditions of de Guenin fail to hold, one proceeds as follows:

(a) Choose a λ and solve [4] for $\phi^*(x, \ )$ and $\phi^{**}(x,\lambda)$ which are the smallest and largest solutions of [4] for a given x and λ, respectively; thus determining A^* and A^{**}.

(b) Next, compute

$$\int_{A^*} \phi^*(x,\lambda)dx = T^* \text{ , and } \int_{A^{**}} \phi^{**}(x,\lambda)dx = T^{**} \text{ ,}$$

if the current λ, say λ_0, is the greatest lower bound on the set λ for which

$$T^* \leq T \leq T^{**},$$

then let

$$\phi_t(x,\lambda_0) = \begin{cases} \phi^*(x,\lambda_0), & x \leq t \\ \phi^{**}(x,\lambda_0), & x > t \end{cases}$$

by the continuity of the integral there exists a t such that

$$\int_c^d \phi_t(x,\lambda_0)dx = T \text{ .}$$

Otherwise change λ, and return to step a[1],

[1] Note that in a discrete search space situation, one may have determined a $\phi_t(x,\lambda_0)$ which satisfies [4] but for which $\sum_x \phi_t(x,\lambda_0) \neq T$. Hence in this case the condition of Zahl is sufficient but not necessary

Appendix D

RANDOM INTERVAL MODELS WITH FINITE VISIBLE PERIODS

D.1 Introduction

This appendix deals with some random interval models with the characteristic that the range of the random variable which denotes the length of the visible period is finite.[1] Some examples are given, their particular properties examined, and finally, a comparison with the Koopman results is made.

D.2 Random Interval of Visibility: Initially Visible

First, it is assumed that the probability density function on the length of the visibility period is given by

$$h(t) = \begin{Bmatrix} \frac{\lambda e^{-\lambda t}}{C} & , & 0 \leq t \leq T_u \\ 0 & , & t > T_u \end{Bmatrix},$$

where $C = 1 - e^{-\lambda T_u}$.

Applying the results of Chapter 3, we obtain for the conditional detection function

[1] If one interprets the length of the visible period as the length of time the target spends in the regions to be searched, such distributions more accurately reflect the search situation.

$$\rho(T) = \frac{1}{C}\left[\frac{k}{\lambda + k}\left(1 - e^{-(\lambda+k)T}\right) - e^{-\lambda T_u}\left(1 - e^{-kT}\right)\right], \quad T \leq T_u,$$
$$\rho(T) = \rho(T_u), \text{ for } T > T_u. \tag{1}$$

Note that if $T_u \to \infty$, one obtains the model of Chapter 2; if, in addition, $\lambda \to 0$, one has the Koopman model.

The First Allocation Rule is readily shown to be

$$p_i k_i = \max_{1 \leq j \leq N} p_j k_j .$$

The conditional detection function is concave over the interval $(0, T_u)$; however, the resulting Kuhn-Tucker conditions don't yield explicit solutions for T. A study of the switching criteria gives the following expression for the switch point (in the 2-region case, assuming the search started in Region 1)

$$T = \frac{1}{k_1}\ln\left(\frac{p_1 k_1}{p_2 k_2 C_1}\right) - \frac{1}{k_1}\ln\left\{\frac{1}{e^{-\lambda_1 T} - e^{-\lambda_1 T_{u_1}}}\right\}, \quad T < T_u \tag{2}$$

An upper limit on the switch time obtained from the above expression is given by

$$\min\left\{T_{u_1}, \left|-\frac{1}{\lambda_1}\left\{\frac{p_2 k_2 C_2}{p_1 k_1} + e^{-\lambda_1 T_{u_1}}\right\}\right|\right\}.$$

Next, we assume that the probability density function on the length of the visibility period is uniform, i.e.,

$$h(t) = \begin{Bmatrix} \frac{1}{S} & 0 < t \leq S \\ 0 & t > S \end{Bmatrix} .$$

Applying the results of Chapter 3, one obtains for the conditional detection function

$$P(T) = 1 - e^{-kT} + \frac{Te^{-kT}}{S} - \frac{1}{kS}\left[1 - e^{-kT}\right], T \leq S \qquad (3)$$

$$P(T) = P(S) \quad \text{for } T > S .$$

Note that as $S \to \infty$, i.e., the target is always visible, one obtains the Koopman model. The First Allocation Rule is identical to the one already given. Again, the conditional detection function is concave on $(0,S)$ and the resulting Kuhn-Tucker conditions fail to yield explicit solutions for T. A study of the switching criteria gives the following expression for the switch point (in the two-region case assuming the search started in Region 1)

$$T = \frac{1}{k_1} \ln\left[\frac{p_1 k_1}{p_2 k_2}\right] - \frac{1}{k_1} \ln\left(1 - \frac{T}{S_1}\right), \; 0 \leq T < S_1 \qquad (4)$$

Since the term

$$-\frac{1}{k_1} \ln\left[1 - \frac{T}{S_1}\right] > 0 ,$$

the solution of the above expression, when it exists, is always larger than the "Koopman Switch Time." Also under conditions $p_1k_1 = p_2k_2$, the Koopman policy splits the effort, but from (4)

$$T = -\frac{1}{k_1}\ln\left(1 - \frac{T}{S_1}\right) .$$

Assuming the search started in Region 1, this implies that once a region is chosen it receives all the effort up to T_s, the solution to (4).

As noted earlier the Kuhn-Tucker conditions don't yield explicit solutions for the optimal allocations; however, one can gain some insight into certain properties of this model by making the following assumptions. In a two-region situation, it is assumed that in the first region the target is always visible (i.e., Koopman model), while in the second region the length of the visible period is uniformly distributed on the interval $(0,S_2)$. Let T denote the total available search time. Then the expression for the amount of effort to be allocated to Region 1 is given by

$$T_1 = \left(\frac{1}{k_1 + k_2}\right)\left\{\ln\left(\frac{p_1 k_1}{p_2 k_2}\right) + k_2 T - \ln\left(1 - \frac{(T - T_1)}{S_2}\right)\right\}. \quad (5)$$

Note that

(a) $(T - T_1) < S_2$, i.e., one never saturates the second region, and

(b) since $-\ln\left(1 - \frac{(T - T_1)}{S_2}\right) \geq 0$ the first region always receives more effort in this mixed situation (more relative to the KSA).

D.3 A Random Interval of Visibility: Random Initiation & Limit

In this section the following assumptions are made:

(1) The random variable corresponding to the start of the visible period is exponentially distributed,

(2) The length of the visibility period is uniformly distributed with parameter S, and

(3) The detection function is exponential.

From the second example of Section D.2 (assuming the visibility period starts at μ and is of length S) the conditional detection function is

$$M(T - \mu) = (1 - e^{-(T-\mu)}) + \left(\frac{T - \mu}{S}\right) e^{-k(T-\mu)} - \frac{1}{kS}(1 - e^{-k(T-\mu)}),$$

for $T - \mu \leq S$, and $M(T - \mu) = M(S)$ for $(T - \mu) > S$.

In Chapter 4, the general expression for the probability of detection for this type of visibility model was given by

$$P\ (t) = \int_0^t f(\mu)M(t - \mu)d\mu\ ,$$

the change of variable $\tau = t - \mu$, yields

$$P\ (t) = \int_0^t f(t - \tau)M(\tau)d\tau\ ,$$

$$M(\tau) = M(S),\ \text{for } \tau > S,\ \text{and for } \tau \le S$$

$$M(\tau) = (1 - e^{-k\tau}) + \frac{\tau}{S} e^{-k\tau} - \frac{1}{kS}(1 - e^{-k\tau})\ .$$

Because of the form of $M(\tau)$, in evaluating $P\ (t)$ one specifies whether $t > S$, or not. Furthermore, special consideration must be given the situation in which $k = \beta$. The density function on the start time is, of course,

$$f(t) = \beta e^{-\beta t},\ t \ge 0\ .$$

Under the above restrictions, one obtains the following expressions for $P\ (t)$.

First for $t \le S$,

(a) $k \ne \beta$

$$P(t) = \left\{1 - \frac{1}{kS}\right\}\left[\left[1 - e^{-\beta t}\right] - \frac{\beta e^{-\beta t}}{k - \beta}\left\{1 - e^{-(k-\beta)t}\right\}\right] \quad (6)$$

$$+ \frac{\beta e^{-\beta t}}{S}\left\{- \frac{te^{-(k-\beta)t}}{(k - \beta)} + \frac{1}{(k - \beta)^2}(1 - e^{-(k-\beta)t})\right\},$$

(b) $k = \beta$,

$$P(t) = (1 - \frac{1}{kS})\left((1 - e^{-\beta t}) - t\beta e^{-\beta t}\right) + \frac{\beta t^2}{2S} e^{-\beta t} \quad (7)$$

Note that by assuming $S \to \infty$, the late arrival cases discussed in Chapter 4 are obtained. Also if $\beta \to \infty$, part (a) tends to the model in Section D.2. For $T > S$, the respective expressions are

(a) $k \neq \beta$

$$P(t) = (1 - \frac{1}{kS})\left\{e^{-\beta(t-S)} - e^{-\beta t}\right\} - (1 - \frac{1}{kS})\frac{\beta e^{-\beta t}}{k - \beta}(1 - e^{-(k-\beta)S})$$

$$+ \frac{1}{S} e^{-\beta t}\left\{\frac{-Se^{-(k-\beta)S}}{(k - \beta)} + \frac{1}{(k - \beta)^2}(1 - e^{-(k-\beta)S})\right\} \quad (8)$$

$$+ (1 - e^{-\beta(t-S)})(1 - \frac{1}{kS}(1 - e^{-kS})) .$$

(b) $k = \beta$

$$p(t) = (1 - \frac{1}{kS})\left\{e^{-\beta(t-S)} - e^{-\beta t}\right\} - (1 - \frac{1}{kS})\, e^{-\beta t}S + \frac{S}{2}\ \beta e^{-\beta t} + (1 - e^{-\beta(t-S)})(1 - \frac{1}{kS}(1 - e^{-kS})) . \tag{9}$$

As $t \to \infty$, a limiting value of P (t) is attained which is equivalent to that obtained in Section D.2, namely

$$P(\infty) = 1 - \frac{1}{kS}(1 - e^{-kS}) .$$

The First Allocation Rules for these models are identical to those given in Chapter 4, i.e., choose as the initial region

$$p_i k_i \beta_i = \max_j \, p_j k_j \beta_j .$$

These functions have the pseudo-concave properties discussed in Chapter 4. Explicit expressions for the optimal allocations cannot be obtained from the Kuhn-Tucker conditions. However, some insight can be gained from the following special case. First, assuming that sufficient total search time is available to insure that either equation 8 or 9 is valid, we observe that the derivatives with respect to time of these expressions are of the form

$$P'(T) = \beta e^{-\beta t} \{\cdot\} , \tag{11}$$

where

$\{\cdot\}$ is a constant with respect to time. The significance of this is that the methods given in Appendix A are applicable and explicit representations of the optimal allocations for the N-region search problem are available. Also, from the approximation discussions of Chapters 3 and 4, the Koopman allocation scheme will be effective for the case $k = \beta$, while for $k \neq \beta$ such may not be true.

The final special case results from the following assumptions. First, consider a two-region case in which $k_1 = \beta_1$, while in the second region, the target is continuously visible, i.e., the standard Koopman situation. Given this scenario with the restriction that $T \leq S_1$, the optimal allocation to the first region is determined from

$$t_1 = \frac{1}{k_1 + k_2} \left\{ \ln \left(\frac{p_1 k_1}{p_2 k_2} \right) + k_2 T \right\}$$

$$+ \frac{1}{k_1 + k_2} \ln \left(\beta_1 \left(T - \frac{t_1^2}{2S_1} \right) \right) , \quad 0 \leq t_1 \leq S_1 . \tag{12}$$

The above expression contains the standard Koopman allocation plus an extra term. Analysis of this term yields the following information.

(a) the expression $f(t) = \beta(t - \frac{t^2}{2S})$ has its maximum at $t = S$, that maximum is $\beta S/2$.

(b) Assuming that T_1 as determined from equation 12 is not on the boundary i.e., $T_1 \neq 0$ or $T_1 \neq S_1$, then

(1) if $f(S) = \beta S/2 < 1$, the optimal allocation is less than the Koopman scheme

(2) if $f(T_1) > 1$, then the reverse is true.

Appendix E

COMPUTER PROGRAMS

E.1 Description and Listing of the Dynamic Programming Computer Program

In this section we shall briefly describe the programs utilized in obtaining the numerical results of the main text and Appendix F. This will be followed by a listing of the FORTRAN level G source statements for the program. First we simply note that the dynamic program for the solution of the problem of maximizing the probability of detection under a constraint on the total available search time uses the standard dynamic programming approach to the constrained maximization problems for each of the models discussed in the text. (See Bellman and Dreyfus (1962)). A glossary of variables unique to our models is given below:

GLOSSARY: Dynamic Programming Source Program

1. Binary Model (MTYPE = 1)

   P(I) = prior probability on target location $1 \leq I \leq 50$.

   PV(I) = prior probability on target visibility in the Ith region, $1 \leq I \leq 50$.

R(I) = the detection rate for the Ith region

2. Random Interval Model: Initial Visibility (MTYPE = 2)

PV(I) = the rate for the length of the visible period in the I^{th} region

R(I) = the detection rate for the I^{th} region

3. Random Interval Model: Random Initiation and Limit (Uniform Distribution on Start Times) (MTYPE = 3)

PV(I) = (See MTYPE = 2)

R(I) = " "

S(I) = the length of the uniform interval on the start of the visible period in region I.

4. Random Interval Model: Random Initiation and Limit (Exponential Start Times) (MTYPE = 4)

PV(I) = (See MTYPE = 2)

R(I) = " "

S(I) = the rate for distribution of starting times for the I^{th} region.

5. General Random Interval Model (MTYPE = 5)

PV(I) = (See MTYPE = 4)

R(I) = " "

S(I) = " "

PHI(I) = prior probability on target visibility at the start of the search in region I.

6. Multiple Interval Model (MTYPE = 6)

PV(I) = (See MTYPE = 5)

R(I) = " "

S(I) = " "

PHI(I) = " "

7. Miscellaneous

NAREA = the number of areas to be searched

DELTA = incremental value of search time (effort) used in the dynamic programming solution of the search problem

TMAX = the total available search time (effort)

PSAVE(I,J) = an intermediate storage array used in the dynamic programming computations

$1 \leq I \leq 500$

$1 \leq J \leq 2$

PSAVE(I,1) = the optimal return

PSAVE(I,2) = the optimal allocation

(These results are printed for each region in this order as they are obtained.)

This program contains a subroutine called DPROB which computes the probability of detection (for any of our models) for a given region with a specified level of search time. The pertinent variables (which were not previously defined) are listed below.

KK = that region for which the probability of detection is to be evaluated (1 < KK < 50)

TAL = the level of search time for which the probability of detection is to be evaluated.

RESULT = the detection probability for region KK when TAL units of search time are allocated to that region.

```
C          MAIN PROGRAM
C           MODEL 1/PV(I)=PROB. TGT. VIS.
C          MODEL 2 / PV(I)=RATE EXP P.D.F. FOR LENGTH VIS. PERIOD
C        MODEL 3 / PV(I)=  RATE "   "       #        "     "
C   MODEL 4 / S(I)=RATE FOR EXP START TIMES
C   MODEL 5 / PHI(I)=PROB. TGT. VIS. COMBINED SINGLE INT. MODEL
C     MODEL 6  MULTIPLE INTERVAL
C      R(I)=RHO=DETECTION RATE
C     PV(I)=LAMBDA=VIS.  P.D.F.
C     S(I)=MU=MASK  P.D.F.
      DIMENSION P(50),PV(50),R(50),PMAX(500,2),PSAVE(500,2)
     1,S(50),PHI(50)
    1 READ(2,1000) MTYPE,NAREA,DELTA,TMAX
      WRITE(3,1002)
      WRITE(3,1000) MTYPE,NAREA,DELTA,TMAX
      GO TO (2,2,2,2,2,2,60),MTYPE
    2 DO 5 I=1,NAREA
    5 READ(2,1001) P(I),PV(I),R(I),S(I),PHI(I)
      DO 6 I=1,NAREA
    6 WRITE(3,1001) P(I),PV(I),R(I),S(I),PHI(I)
C          INITIALIZE REGIONAL DETECTION PROBABILITY SUBROUTINE
      CALL DPROB1(MTYPE,NAREA,P,PV,R,S,PHI)
      NCHK=(TMAX/DELTA)+.5
      IF (NCHK.GT.500) GO TO 100
C          ZERO STARTING VALUES OF OBJ. FCN. AND CORRS. ALLOC.
      NZERO=NCHK+1
      DO 7 J=1,2
      DO 7 I=1,NZERO
    7 PMAX(I,J)=0.
C          SET REGIONAL COUNTER
      K=1
C          INITIALIZE TOTAL AVAILABLE SEARCH TIME
   10 TNOW=0.
C          SET CURRENT MAXIMUM
   15 BETA=-1.E-05
C          INITIALIZE CURRENT VALUE OF SEARCH TIME
      TK=0.
   20 INDEXC=((TNOW-TK)/DELTA)+0.5+1
C         OBTAIN DETECTION PROB FOR CURRENT ALLOC.
      CALL DPROB(K,TK,GT)
      IF(GT.LT.0.) GT=0.
      ALPHA=GT+PMAX(INDEXC,1)
C          TEST FOR NEW MAXIMUM
      IF(ALPHA.LT.BETA) GO TO 30
      BETA=ALPHA
      GAMMA=TK
   30 TK=TK+DELTA
      IF (TK.GT.TNOW) GO TO 40
      GO TO 20
```

```
C          SAVE NEW MAX. & CORRES. ALLOCATION
   40 INDEXS=(TNOW/DELTA)+0.5+1
      PSAVE(INDEXS,1)=BETA
      PSAVE(INDEXS,2)=GAMMA
      TNOW=TNOW+DELTA
      IF (TNOW.GT.TMAX) GO TO 50
      GO TO 15
C          PRINT RESULTS FOR THIS REGION
   50 WRITE(3,1002) K
      NPRINT=NCHK+1
      DO 51 I1=1,NPRINT

   51 WRITE(3,1003) PSAVE(I1,1),PSAVE(I1,2)
C          TRANSFER P9EVIOUS RESULTS
      DO 55 J=1,2
      DO 55 I=1,NPRINT
   55 PMAX(I,J)=PSAVE(I,J)
      K=K+1
C          TEST FOR MORE REGIONS
      IF (K.GT.NAREA) GO TO 1
      GO TO 10
C          ERROR RETURNS
  100 WRITE(3,1004)
   60 CALL EXIT
 1000 FORMAT(2I2,6X,3F10.5)
 1001 FORMAT(5F10.5)
 1002 FORMAT(1H1,I3)
 1003 FORMAT(10X,F10.5,5X,F10.5)
 1004 FORMAT(1H1,5X,' ERROR IN MODEL SPECIFICATIONS ' )
      END
      SUBROUTINE DPROB1(MTYPE,NAREA,P,PV,R,S,PHI)
      DIMENSION P(50),PV(50),RATE(50),PVIS(50),R(50),PL(50)
     1,S(50),SR(50),PHI(50),SPHI(50)
      REAL NUM1
C          SUBRT. CALC. DET. PROB./ AN ALLOC. OF TIME(TAL)
      DO 5 I=1,NAREA
      PL(I)=P(I)
      PVIS(I)=PV(I)
      SR(I)=S(I)
      SPHI(I)=PHI(I)
    5 RATE(I)=R(I)
      RETURN
      ENTRY DPROB (KK,TAL,RESULT)
      IF(TAL.GT.0.) GO TO 7
      RESULT=0.
      RETURN
    7 GO TO (10,20,20,40,20,60),MTYPE
C          MODEL 1 BINARY INTERVISIBILITY
   10 RESULT=PL(KK)*PVIS(KK)*(1.-EXP(-RATE(KK)*TAL))
      RETURN
```

```
C           MODEL 2 SINGLE INTERVAL,START AT TIME ZERO
   20 RESULT=PL(KK)*(RATE(KK)/(RATE(KK)+PVIS(KK)))
      IF (MTYPE.EQ.3) GO TO 30
      RESULT=RESULT*(1.-EXP(-(RATE(KK)+PVIS(KK))*TAL))
      IF(MTYPE.EQ.5) GO TO 50
      RETURN
C           MODEL 3 SINGLE INTERVAL UNIFORM START TIME
C         SR(KK) DENOTES LENGTH OF UNIFORM INT. ON START TIME
   30 IF (TAL.NE.0.) GO TO 31
      RESULT=0.
      RETURN
   31 IF(TAL.GT.SR(KK)) GO TO 310
      TEMP=1.-EXP(-(RATE(KK)+PVIS(KK))*TAL)
      TEMP=TEMP/(RATE(KK)+PVIS(KK))
      RESULT=RESULT*(TAL-TEMP)/SR(KK)
      RETURN
  310 TEMP=EXP(-(RATE(KK)+PVIS(KK))*TAL)
      TEMP=EXP(-(RATE(KK)+PVIS(KK))*(TAL-SR(KK)))-TEMP
      TEMP=SR(KK)-TEMP/(RATE(KK)+PVIS(KK))
      RESULT=RESULT*TEMP/SR(KK)
      RETURN
C     MODEL 4 SINGLE INT. EXP START TIME

C     PV(I)=RATE FOR VIS. DIST
C     RATE(I)=RATE FOR COND. DETECT. FUNCT.
C     SR(I)=RATE FOR START VIS. PERIOD
   40 DEN=SR(KK)-RATE(KK)-PVIS(KK)
      NUM1=RATE(KK)+PVIS(KK)
      IF(ABS(DEN).LT.1.E-4) GO TO 410
      RESULT=SR(KK)*EXP(-NUM1*TAL)-NUM1*EXP(-SR(KK)*TAL)
      RESULT=PL(KK)*RATE(KK)*(1.-RESULT/DEN)/NUM1
      IF(MTYPE.EQ.5) GO TO 52
      RETURN
  410 TEMP=EXP(-SR(KK)*TAL)*(1.+SR(KK)*TAL)
      RESULT=PL(KK)*RATE(KK)*(1.-TEMP)/NUM1
      IF(MTYPE.EQ.5) GO TO 52
      RETURN
C     MODEL 5 COMBO. ZERO &NON-ZERO START TIMES
   50 SAVE1=RESULT
      GO TO 40
   52 RESULT=SPHI(KK)*SAVE1+(1.-SPHI(KK))*RESULT
      RETURN
C     MODEL 6 MULTIPLE INTERVAL
C      SOLVE FOR ROOTS
   60 B=SR(KK)+PVIS(KK)+RATE(KK)
      C=SR(KK)*RATE(KK)
      A=1.
      DISC=B*B-4.*A*C
      IF(DISC.LT.0.) GO TO 100
      ROOT2=(B+SQRT(DISC))/(2.*A)
```

```
      ROOT1=(B-SQRT(DISC))/(2.*A)
      DROOT=ROOT2-ROOT1
C     COMPUTE ABSORPTION PROB
C---------P23(T)
      TEMP1=(1.-EXP(-TAL*ROOT1))/ROOT1
      TEMP2=(1.-EXP(-TAL*ROOT2))/ROOT2
      TEMP=SR(KK)*RATE(KK)*(TEMP1-TEMP2)/DROOT
C-------P13(T)
      SAVE=RATE(KK)*(SR(KK)-ROOT1)*TEMP1/DROOT
      SAVE=SAVE-(SR(KK)-ROOT2)*TEMP2*RATE(KK)/DROOT
C        COMPUTE STATE PROB.
      RESULT=TEMP*(1.-SPHI(KK))
      RESULT=RESULT+SAVE*SPHI(KK)
      RESULT=PL(KK)*RESULT
      RETURN
  100 RESULT=-20.
      RETURN
      END
```

E.8 *Description and Listing of the Computer Program for the Comparison of the Optimal Search Policy with the Associated Koopman Policy*

Here the program used in the determination of Section F.4 of Appendix F is briefly described. This is followed by a listing of the FORTRAN level G source statements for this program. The approach taken may be summarized as follows:

1. The dynamic programming routine described in Section E.1 is used to generate the optimal search policy, region by region.
2. When the results for the last region have been attained, one then computes the allocations under the Koopman policy.[1]
3. The difference in the return from the two policies is then computed as well as the percent relative error for each level of search time up to the maximum specified level.

Since this program makes use of the one just described, we shall list in the following glossary only those variables which were not previously defined.

[1]Here we are restricted to two regions; however, this restriction can easily be removed by extending the Koopman calculations to N-regions via the techniques described in Appendix A.

GLOSSARY:

LPRNT, IBACK = Flags used to control the printing of the difference curve, i.e.,

LPRNT \ IBACK	= 0	≠ 0
= 0	peak only	peak up to T_{max}
≠ 0	up to the peak	complete curve

ITEST = Flag to indicate that the computation is to be continued until the percent relative error is less than PERROR

PERROR = Limiting value of the percent relative error

Output Description

(a) Field 1 = optimal return

Field 2 = return under the Koopman policy

Field 3 = difference (1-2)

Field 4 = allocation to region 1 under the Koopman policy

Field 5 = allocation to region 2 under the Koopman policy

Field 6 = total available search time

(b) If IBACK = 0 or ITEST = 0,

Field 1 = difference in the optimal return (optimal-Koopman)

Field 2 = total available search time

Field 3 = percent relative error at the peak difference

Field 4 = optimal return (probability of detection)

Field 5 = Koopman expected time until detection

Field 6 = Koopman switch time.

```
C          MAIN PROGRAM
C           MODEL 1/PV(I)=PROB. TGT. VIS.
C          MODEL 2 / PV(I)=RATE EXP P.D.F. FOR LENGTH VIS. PERIOD
C        MODEL 3 / PV(I)=  RATE "   "        #        "    "
C   MODEL 4 / S(I)=RATE FOR EXP START TIMES
C   MODEL 5 / PHI(I)=PROB. TGT. VIS. COMBINED SINGLE INT. MODEL
C     MODEL 6  MULTIPLE INTERVAL
C      R(I)=RHO=DETECTION RATE
C     PV(I)=LAMBDA=VIS.  P.D.F.
C     S(I)=MU=MASK  P.D.F.
      DIMENSION P(50),PV(50),R(50),PMAX(500,2),PSAVE(500,2)
     1,S(50),PHI(50),XL(8)
    1 READ(2,1000) MTYPE,NAREA,LPRNT,IBACK,ITEST,DELTA,TMAX,PERROR
      WRITE(3,1000) MTYPE,NAREA,LPRNT,IBACK,ITEST,DELTA,TMAX,PERROR
      GO TO (2,2,2,2,2,2,60),MTYPE
    2 DO 5 I=1,NAREA
    5 READ(2,1001) P(I),PV(I),R(I),S(I),PHI(I)
      DO 6 I=1,NAREA
    6 WRITE(3,1001) P(I),PV(I),R(I),S(I),PHI(I)
C          INITIALIZE REGIONAL DETECTION PROBABILITY SUBROUTINE
      CALL DPROB1(MTYPE,NAREA,P,PV,R,S,PHI)
      SAVE1=-1.E+05
      SAVE2=0.
      SAVE3=0.
      SAVE4=0.
      IOVER=0
      IEXIT=0
  496 Z1=ALOG(P(1)*R(1)/(P(2)*R(2)))
      Z2=Z1/R(1)
      IF(Z2.LT.0.) GO TO 5100
      IFIRST=1
      GO TO 66
 5100 Z2=-Z1/R(2)
      IFIRST=2
```

```
   66 CALL  TKOOP(P(1),P(2),R(1),R(2),SAVE5)
      NCHK=(TMAX/DELTA)+.5
      IF (NCHK.GT.500) GO TO 100
C          ZERO STARTING VALUES OF OBJ. FCN. AND CORRS. ALLOC.
      NZERO=NCHK+1
      DO 7 J=1,2
      DO 7 I=1,NZERO
    7 PMAX(I,J)=0.
C          SET REGIONAL COUNTER
      K=1
C          INITIALIZE TOTAL AVAILABLE SEARCH TIME
   10 TNOW=0.
C          SET CURRENT MAXIMUM
   15 BETA=-1.E-05
C          INITIALIZE CURRENT VALUE OF SEARCH TIME
      TK=0.
      IF(K.EQ.1) TK=TNOW
   20 INDEXC=((TNOW-TK)/DELTA)+0.5+1
C          OBTAIN DETECTION PROB FOR CURRENT ALLOC.
      CALL DPROB(K,TK,GT)
      IF (GT.LT.0.) GO TO 110
      ALPHA=GT+PMAX(INDEXC,1)
C          TEST FOR NEW MAXIMUM
      IF(ALPHA.LT.BETA) GO TO 30
      BETA=ALPHA

      GAMMA=TK
   30 TK=TK+DELTA
      IF (TK.GT.TNOW) GO TO 40
      GO TO 20
C          SAVE NEW MAX. & CORRES. ALLOCATION
   40 INDEXS=(TNOW/DELTA)+0.5+1
      PSAVE(INDEXS,1)=BETA
      PSAVE(INDEXS,2)=GAMMA
      IF(K.EQ.NAREA) GO TO 499
   45 TNOW=TNOW+DELTA
      IF (TNOW.GT.TMAX) GO TO 50
      GO TO 15
C          PRINT RESULTS FOR THIS REGION
   50 NPRINT=NCHK+1
      IF(LPRNT.EQ.0) GO TO 52
      WRITE(3,1002) K
      DO 51 I1=1,NPRINT
   51 WRITE(3,1003) PSAVE(I1,1),PSAVE(I1,2)
C          TRANSFER PREVIOUS RESULTS
```

```
   52 CONTINUE
      DO 55 J=1,2
      DO 55 I=1,NPRINT
   55 PMAX(I,J)=PSAVE(I,J)
   57 K=K+1
C          TEST FOR MORE REGIONS
      IF(K.GT.NAREA) GO TO 505
      GO TO 10
C          ERROR RETURNS
  100 WRITE(3,1004)
      GO TO 60
  110 WRITE(3,1005)
   60 CALL EXIT
C------ COMPUTE RESULTS FROM KOOPMAN SCHEME
  499 CONTINUE
      IF(TNOW.GT.Z2) GO TO 5012
      GO TO (5008,5009),IFIRST
 5008 T1=TNOW
      T2=0.
      GO TO 5014
 5009 T2=TNOW
      T1=0.
      GO TO 5014
 5012 T1=(Z1+R(2)*TNOW)/(R(1)+R(2))
      T2=(-Z1+R(1)*TNOW)/(R(1)+R(2))
 5014 IF(MTYPE.LT.3) GO TO 100
      K=1
      CALL DPROB(K,T1,G1)
      K=2
      CALL DPROB(K,T2,G2)
      PROB=G1+G2
      TEMP=PSAVE(INDEXS,1)-PROB
      IF((ITEST.EQ.0).OR.(IOVER.EQ.0)) GO TO 5007
      IF(PSAVE(INDEXS,1).EQ.0.) GO TO 5007
      TEST=TEMP/PSAVE(INDEXS,1)
      IF(TEST.GT.PERROR) GO TO 5007
      IEXIT=1
      GO TO 5013
 5007 IF((IOVER.EQ.1).AND.(IBACK.NE.0)) GO TO 5013
      IF(LPRNT.EQ.0) GO TO 5015
 5013 WRITE(3,1001) PSAVE(INDEXS,1),PROB,TEMP,T1,T2,TNOW

      IF(IEXIT.EQ.1) GO TO 1
 5015 IF(TEMP.LT.0.) GO TO 500
      IF(TEMP.LT.SAVE1) GO TO 510
      SAVE1=TEMP
      SAVE2=TNOW
      IF(PSAVE(INDEXS,1).EQ.0.) GO TO 500
      SAVE3=TEMP/PSAVE(INDEXS,1)
      SAVE4=PSAVE(INDEXS,1)
```

```
  500 GO TO 45
  505 IF((IBACK.NE.0).OR.(ITEST.NE.0)) GO TO 507
      WRITE(3,1006)
      WRITE(3,1001) SAVE1,SAVE2,SAVE3,SAVE4,SAVE5,Z2
      GO TO 1
  507 WRITE(3,1007) PERROR
      WRITE(3,1001) PSAVE(INDEXS,1),PROB,TEMP,T1,T2,TNOW
      GO TO 1
  510 CONTINUE
      IF(IOVER.EQ.1) GO TO 511
      WRITE(3,1001) SAVE1,SAVE2,SAVE3,SAVE4,SAVE5,Z2
      IOVER=1
  511 IF((IBACK.NE.0).OR.(ITEST.NE.0)) GO TO 45
      GO TO 1
 1000 FORMAT(5I2,3F10.5)
 1001 FORMAT(8F10.5)
 1002 FORMAT(1H1,I3)
 1003 FORMAT(10X,F10.5,5X,F10.5)
 1004 FORMAT(1H1,5X,' ERROR IN MODEL SPECIFICATIONS ' )
 1005 FORMAT(1H1,5X,' ERROR IN DET. PROB. SUBROUTINE' )
 1006 FORMAT(' DIFFERENCE FCN STILL INCREASING')
 1007 FORMAT('PERCENT RELATIVE ERROR LARGER THAN  ',F10.5)
      END
      SUBROUTINE DPROB1(MTYPE,NAREA,P,PV,R,S,PHI)
      DIMENSION P(50),PV(50),RATE(50),PVIS(50),R(50),PL(50)
     1,S(50),SR(50),PHI(50),SPHI(50)
      REAL NUM1
C          SUBRT. CALC. DET. PROB./ AN ALLOC. OF TIME(TAL)
      DO 5 I=1,NAREA
      PL(I)=P(I)
      PVIS(I)=PV(I)
      SR(I)=S(I)
      SPHI(I)=PHI(I)
    5 RATE(I)=R(I)
      RETURN
      ENTRY DPROB (KK,TAL,RESULT)
      GO TO (10,20,20,40,20,60),MTYPE
C           MODEL 1 BINARY INTERVISIBILITY
   10 RESULT=PL(KK)*PVIS(KK)*(1.-EXP(-RATE(KK)*TAL))
      RETURN
C           MODEL 2 SINGLE INTERVAL,START AT TIME ZERO
   20 RESULT=PL(KK)*(RATE(KK)/(RATE(KK)+PVIS(KK)))
      IF (MTYPE.EQ.3) GO TO 30
      RESULT=RESULT*(1.-EXP(-(RATE(KK)+PVIS(KK))*TAL))
      IF(MTYPE.EQ.5) GO TO 50
      RETURN
C           MODEL 3 SINGLE INTERVAL UNIFORM START TIME
C        SR(KK) DENOTES LENGTH OF UNIFORM INT. ON START TIME
   30 IF (TAL.NE.0.) GO TO 31
      RESULT=0.
      RETURN
   31 IF(TAL.GT.SR(KK)) GO TO 310
```

```
      TEMP=1.-EXP(-(RATE(KK)+PVIS(KK))*TAL)
      TEMP=TEMP/(RATE(KK)+PVIS(KK))
      RESULT=RESULT*(TAL-TEMP)/SR(KK)
      RETURN
  310 TEMP=EXP(-(RATE(KK)+PVIS(KK))*TAL)
      TEMP=EXP(-(RATE(KK)+PVIS(KK))*(TAL-SR(KK)))-TEMP
      TEMP=SR(KK)-TEMP/(RATE(KK)+PVIS(KK))
      RESULT=RESULT*TEMP/SR(KK)
      RETURN
C     MODEL 4 SINGLE INT. EXP START TIME
C     PV(I)=RATE FOR VIS. DIST
C     RATE(I)=RATE FOR COND. DETECT. FUNCT.
C     SR(I)=RATE FOR START VIS. PERIOD
   40 DEN=SR(KK)-RATE(KK)-PVIS(KK)
      NUM1=RATE(KK)+PVIS(KK)
      IF(ABS(DEN).LT.1.E-4) GO TO 410
      RESULT=SR(KK)*EXP(-NUM1*TAL)-NUM1*EXP(-SR(KK)*TAL)
      RESULT=PL(KK)*RATE(KK)*(1.-RESULT/DEN)/NUM1
      IF(MTYPE.EQ.5) GO TO 52
      RETURN
  410 TEMP=EXP(-SR(KK)*TAL)+SR(KK)*TAL*EXP(-NUM1*TAL)
      RESULT=PL(KK)*RATE(KK)*(1.-TEMP)/NUM1
      IF(MTYPE.EQ.5) GO TO 52
      RETURN
C     MODEL 5 COMBO. ZERO &NON-ZERO START TIMES
   50 SAVE1=RESULT
      GO TO 40
   52 RESULT=SPHI(KK)*SAVE1+(1.-SPHI(KK))*RESULT
      RETURN
C     MODEL 6 MULTIPLE INTERVAL
C      SOLVE FOR ROOTS
   60 B=SR(KK)+PVIS(KK)+RATE(KK)
      C=SR(KK)*RATE(KK)
      A=1.
      DISC=B*B-4.*A*C
      IF(DISC.LT.0.) GO TO 100
      ROOT2=(B+SQRT(DISC))/(2.*A)
      ROOT1=(B-SQRT(DISC))/(2.*A)
      DROOT=ROOT2-ROOT1
C     COMPUTE ABSORPTION PROB
C---------P23(T)
      TEMP1=(1.-EXP(-TAL*ROOT1))/ROOT1
      TEMP2=(1.-EXP(-TAL*ROOT2))/ROOT2
      TEMP=SR(KK)*RATE(KK)*(TEMP1-TEMP2)/DROOT
C-------P13(T)
      SAVE=RATE(KK)*(SR(KK)-ROOT1)*TEMP1/DROOT
      SAVE=SAVE-(SR(KK)-ROOT2)*TEMP2*RATE(KK)/DROOT
```

```
C         COMPUTE STATE PROB.
      RESULT=TEMP*(1.-SPHI(KK))
      RESULT=RESULT+SAVE*SPHI(KK)
      RESULT=PL(KK)*RESULT
      RETURN
  100 RESULT=-20.
      RETURN
      END
C----- COMPUTE EXPT. TIME TO DETECT KOOPMAN MODEL
      SUBROUTINE TKOOP(X1,X2,Y1,Y2,EC)
      P1=X1
      P2=X2
      R1=Y1
      R2=Y2
      R=P1*R1/(P2*R2)
      Q=(R1+R2)/(R1*R2)
      IF(R.EQ.1.) GO TO 15
      IF(R.GT.1.) GO TO 10
C------ INTERCHANGE PARAMETERS
      SAVE1=P1
      SAVE2=R1
      P1=P2
      R1=R2
      P2=SAVE1
      R2=SAVE2
   10 R=P1*R1/(P2*R2)
      T=ALOG(R)/R1
      Z=(R1+R2)/R2
      EC=Z*Q*EXP(-T/Q)
      EC=EC*R**(-(1.-1./Z))+(1.-EXP(-R1*T))/R1
      EC=EC*P1+P2*T
      GO TO 20
   15 EC=0
   20 RETURN
      END
```

Appendix F

NUMERICAL COMPARISON WITH THE KOOPMAN MODEL

F.1 Introduction

In this appendix we investigate the situation in which a partially informed searcher being aware of the earlier results of Koopman (which assume continuously visible targets) applies them to situations in which the target behavior is actually characterized by one of the intervisibility processes. The consequences (in terms of the probability of detecting the target under a constraint on the total available search effort) of this application of the Koopman results will be compared to those obtained from the optimal allocation procedures for these models. In this appendix these comparisons are made from the results of a numerical analysis, rather than the approximation analyses of Chapters 4 and 5.

This method of comparison has a great deal of practical significance (in addition to being a useful mathematical device for comparing the two models). For example, one may view each of the models under study as being composed of two types of parameters: those which the searcher may be expected to readily obtain or generate, e.g., the prior probabilities on target location, and the regional detection

capabilities; and those which essentially characterize the visibility processes, e.g., the distributions on the lengths of the masked and visible periods. The former group may be labeled the "searcher" parameters; the latter, the "target behavior/environmental" parameters. It is not unreasonable to assume that the searcher may

(a) have imperfect knowledge of "target behavior/environmental" parameters, and

(b) not be able to obtain estimates of these parameters or be unaware of the fact that the target is not continuously visible.[1]

F.2 Description of the Technique

In making a comparison such as this, the decision maker might generate the following list of questions relative to the appropriateness of the Koopman policy (denoted SKA) in situations in which the target behavior is characterized by one of the visibility processes.

(1) Under what conditions, if any, does the use of the SKA policy lead to small errors? By small errors, the decision maker may mean a relative error < 5%, at the peak difference in the probability of detection.

(2) Which, if any, of the following combinations of "searcher" and "target" parameters yield small errors?

[1]In Section 2.2.3, we discuss, conceptually, the implications of this analysis in terms of the various options open to the searcher as well as the associated cost-effectiveness measures.

Searcher	Identical regional detection capability Non-identical regional detection capability
Target	Identical regional behavior/environment Non-identical regional behavior/environment

(3) In those situations in which the mean time to detect is much less than the mean length of the visible period, can the searcher safely assume the SKA to be adequate?

(4) In those situations in which the SKA is shown to be inadequate, are there any simple relationships between the detection and target behavior parameters which indicate this inadequacy?

These questions apply to two situations: a *short-term* search situation in which the total available search time is less than the Koopman expected time for detection; and, a *long-term* situation in which the decision maker has an amount of time much greater than the Koopman expected time.

In order to answer any of these questions, one has to decide on certain measures of the effectiveness of using the SKA. As a measure of the worst case situation in using the SKA, consider

(a) the percent relative error at the peak difference in the probability of detection (% rel.).

However, the decision-maker also needs to know at what level of search effort this peak occurs in order to judge its

significance relative to the total amount of search time he has available. Thus we take as our second measure,

(b) that level of total available search time at which the peak difference occurs (Peak Time).

The analysis of the models of Chapters 2 and 3 indicated that the error resulting from the SKA could be reduced by increasing the total available search time. Thus the decision maker (searcher) is interested in determining

(c) the level of total available search time at which the percent relative error is less than some specified value.

Assuming that the searcher's strategy is to allocate an amount of time *at least* equal to the expected time to detect the target under the Koopman strategy, the following results might occur: the percent relative error is moderate but the peak time is much greater than the Koopman expected time; or the percent relative error may be high and occur at, or before, the Koopman expected time. Both of these cases would appear to be significant to the decision maker. The following measure is used to describe these situations:

(d) $$R = \frac{\text{Peak Time}}{\text{KET}} \cdot (\%\ \text{rel.}) .$$

In a certain sense, the Koopman expected time may be used to characterize the classical search situation as to the

degree of difficulty. Thus it was chosen to normalize the time at which the peak differences occur.

The limited analytical results for the models of Chapters 4 and 5 provide "explicit" answers to the above questions in very specific cases. In particular, the approximations (valid for large quantities of search effort) yield analytical expressions which can be studied to provide the required answers. Situations in which the available effort is limited to the extent that under the optimal policy all the effort is placed in a single region will also lend themselves to such analysis. However, in order to study the error function over the spectrum of search time, numerical techniques are required. Since this type of problem is readily solved using dynamic programming, this procedure is used in the following numerical analysis.

Having obtained the optimal allocation policy and return for a given level of search effort, the return from the SKA at this level of effort and the difference in the two returns is computed. If interest centers on the peak differences, as soon as the difference function begins to decrease the computation is ended. If one is interested in determining the amount of effort required to reduce the

percent relative error below some specified value, the computation continues until that point.[1] In the peak difference situation, the nature of a dynamic programming solution requires that we only compute the optimal solution for the last region until the peak is passed. A flow chart for these computations is given in Figure F.1.

F.3 Summary of the Numerical Results

The charts and tables given in this section contain the results of a series of computer runs for each of the models represented in Figure F.1. Because the dimensionality required to describe the search and target-environment parameter space is extremely large, only the two-region case is considered. Figure F.2 is a schematic representation of the parameter ranges used.

A summary of the results for each model relative to the questions formulated in Section F.2 is given. These results are identified by the ratio of prior probabilities on target location and the values of the detection rates for each region. Finally, a series of tables containing the measures introduced in Section F.2 are given.

[1] The unimodality of the error function is determined via an analysis of the approximate error functions given for each of the models.

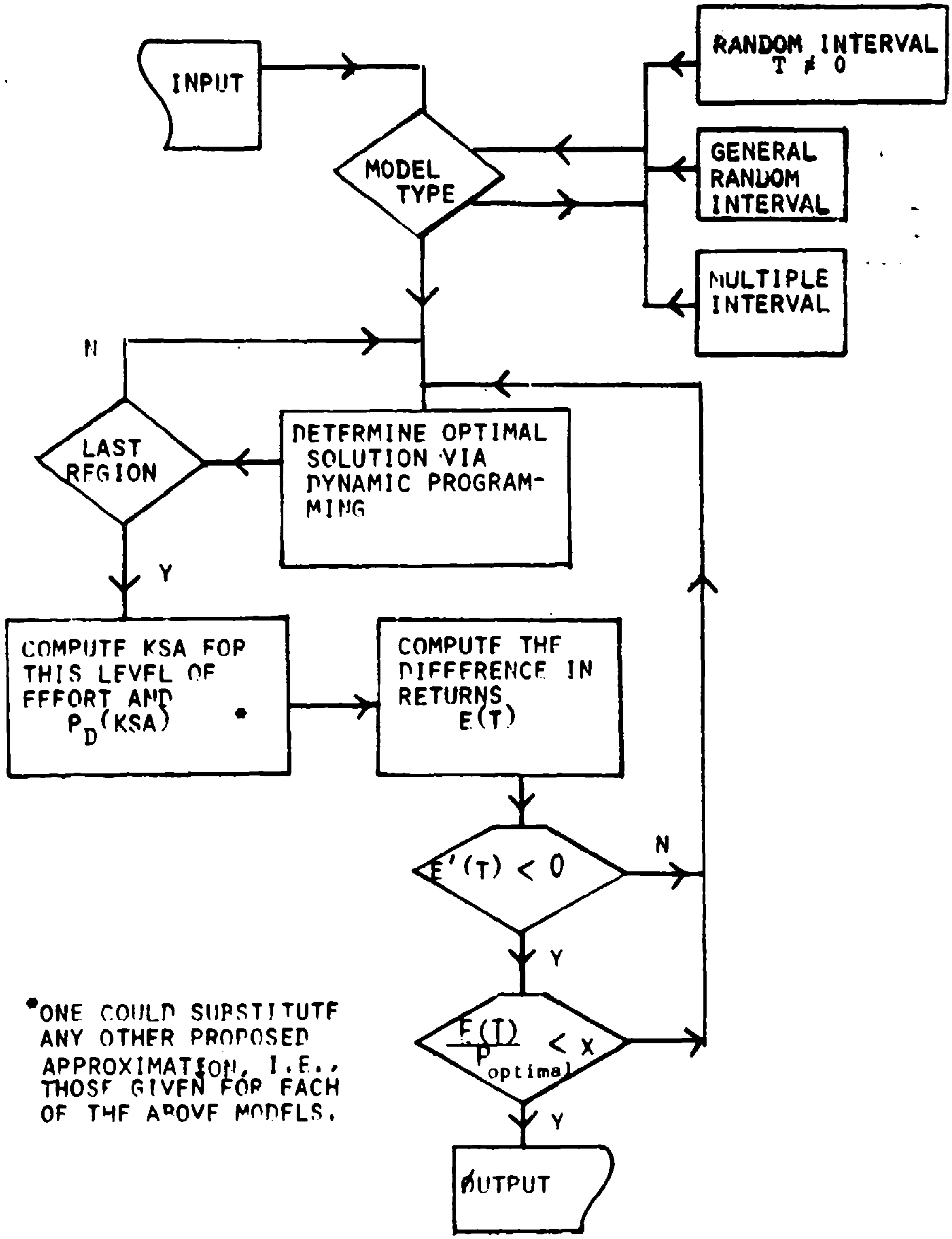


Figure F.1 Sensitivity Computations

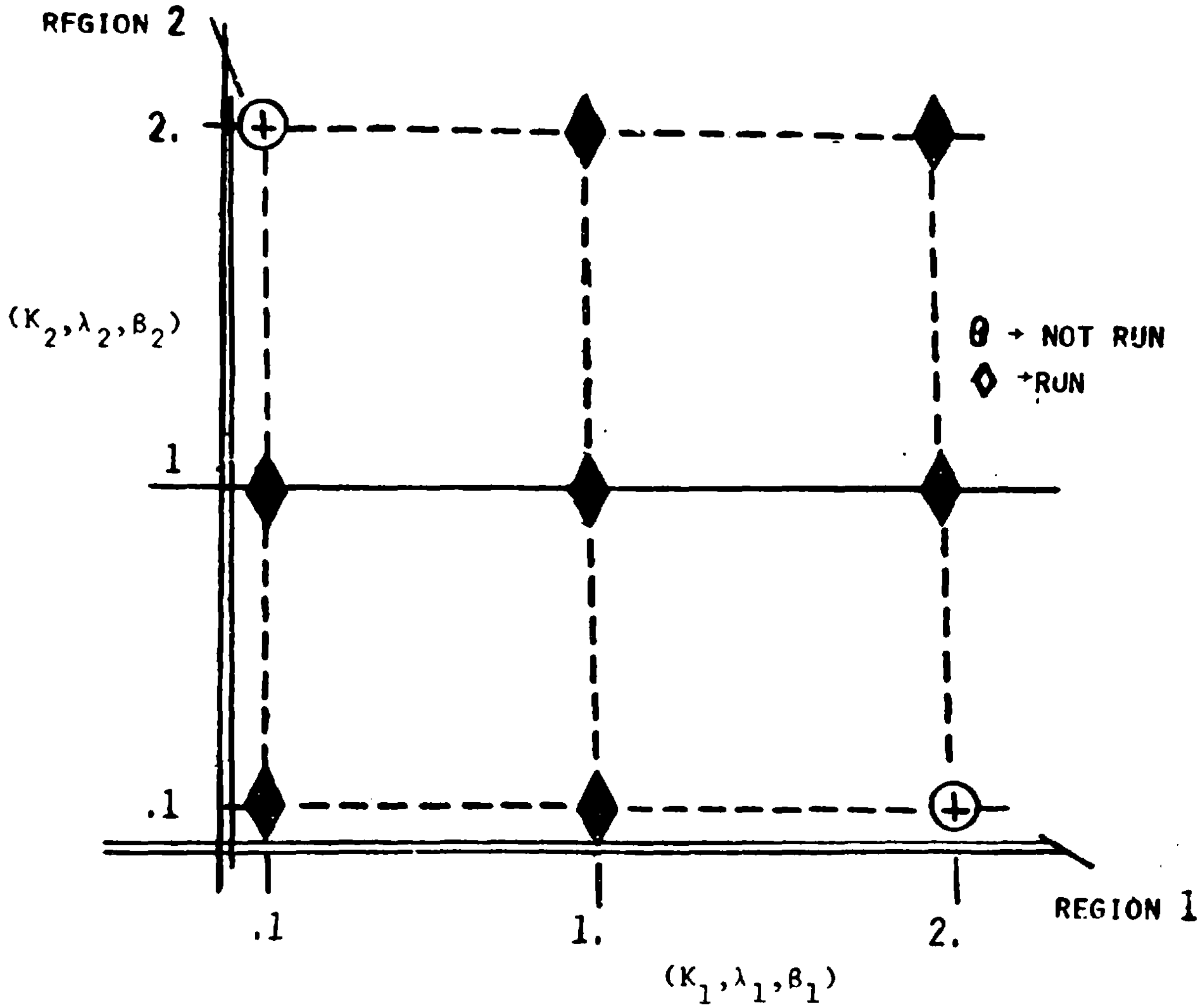


Figure F.2 Visibility Parameter Space

(I) Random Interval/Random Initiation and Limit

Exponential distributions are assumed on the start time as well as on the length of the visible period and the detection devices. The following results were obtained. The labeling corresponds to the respective question number of Section F.2.

(1) The availability of search time on the order of 2-KET (Koopman expected time) will *not* reduce the relative error below 5% in using the SKA.

(2) In those cases in which the mean time to detect is much less than the mean length of the visibility period, large errors are obtained from the use of SKA.

(3) For the short-term search situation:

$$p_1/p_2 = 1$$

k_1 \ k_2	0.1	1.0
0.1	Min % R.E. = 21[1]	See tables
1.0	See table[2] F.5	Min % R.E. = 19

[1] Over the visibility parameter space under investigation, none of the combinations studied led to small short-term errors in using the Koopman policy.

[2] Note, in many of these cases, the extreme sensitivity of the error function measures to changes in the visibility parameters.

$p_1/p_2 = 2$

k_1 \ k_2	0.1	1.0
0.1	See table F.6	Min % R.E. = 11
1.0	See table F.6	Min % R.E. = 16

(4) From the tables, we observe that the greatest errors in using the SKA occur with non-identical detectors and that level of total available search effort at which they occur is, in many cases, much greater than the Koopman expected time (KET). The approximation analysis of Section 4.2.2 gives insight into the causes of such sustained errors. Consider the following example:

p	λ	k	β
2/3	.1	1.	.1
1/3	1	.1	.1

From the analysis of Section 4.2.2, the long-term approximate policies are

$$t_1 = \frac{\beta_2 t}{\beta_1 + \beta_2} \quad \text{and} \quad t_2 = \frac{\beta_1 t}{\beta_1 + \beta_2},$$

while the similar results for the Koopman policy are

$$t_1 = \frac{k_2}{k_1 + k_2}\, t \quad \text{and} \quad t_2 = \frac{k_1 t}{k_1 + k_2}.$$

Thus, under the Koopman policy region 1 doesn't receive enough of the search effort. Also note that if $\beta_1 = 1$, then the long-term policies are identical under both methods.[1] To summarize, an investigation of the relative magnitudes of the conditional detection rate $(\lambda_i + k_i)$ and the rate of masking interval, β_i, determines the long-term allocation policy. When this policy greatly differs from the Koopman policy, the latter will be inadequate over a large range of total available search effort.

Tables F.1 and F.2 summarize the main effects due to the various search parameters; measure (d) is used in the tables. The following observations are of interest.

(a) For $k_1 = k_2 = .1$, note the increasing sensitivity of measure (d) to λ_1 and λ_2 with increasing prior positional probability ratios p_1/p_2.

(b) For $k_1 = .1$, and $k_2 = 1.$, note the sensitivity to β_2 and the reversal of its effect at $p_1/p_2 = 10$.

(c) For $k_1 = 1.$, $k_2 = .1$, note the *extreme* sensitivity of measure (d) to β_1. If one uses the approximate solutions of Section 4.2.2, then it is apparent that at the lower level of β_1 the Koopman and optimal policies are very different.

(d) For $k_1 = 1. = k_2$, we note the increasing sensitivity to β_1.

[1] Which explains the senstivity to β_1 in the tables.

Table F.1

Random Interval Sensitivity

$p_1/p_2=1$		$k_1 = .1$, $k_2 = .1$	$k_1 = .1$, $k_2 = 1.$	$k_1 = 1.$, $k_2 = .1$	$k_1 = 1.$, $k_2 = 1.$
$\lambda_1 =$	.1	92.9	149.6	570.3	315.5
	1.	71.3	945.8	525.2	291.
$\lambda_2 =$	.1	92.9	569.9	149.7	315.5
	1.	71.3	525.5	945.8	291.
$\beta_1 =$	.1	115.1	582.9	1035.5	355.5
	1.	52.7	513.	60.	251.
$\beta_2 =$	.1	112.5	1035.5	589.2	355.5
	1.	54.6	60.2	506.3	251.
$p_1/p_2=2$					
$\lambda_1 =$	.1	63.8	103.8	1137.9	540.6
	1.	168.3	485.2	662.5	193.7
$\lambda_2 =$	.1	167.8	306.6	588.6	445.
	1.	64.3	255.4	1211.9	289.3
$\beta_1 =$	.1	131.5	497.2	1747.1	522.4
	1.	100.6	64.8	53.4	211.9
$\beta_2 =$	.1	138.4	491.7	812.7	342.3
	1.	93.7	70.3	987.8	392.

Table F.2

Random Interval Sensitivity

$p_1/p_2=3$		$k_1 = .1$ $k_2 = .1$	$k_1 = .1$ $k_2 = 1.$	$k_1 = 1.$ $k_2 = .1$	$k_1 = 1.$ $k_2 = 1.$
$\lambda_1 =$	.1	49.2	91.6	1418.6	492.1
	1.	195.6	431.6	1197.7	339.5
$\lambda_2 =$	.1	188.2	291.2	1143.4	390.4
	1.	56.5	232.0	1472.9	44.2
$\beta_1 =$	.1	124.4	457.4	2566.0	639.4
	1.	120.3	65.8	56.5	191.4
$\beta_2 =$	.1	135.2	447.8	1090.8	559.9
	1.	109.5	75.4	1525.7	270.9
$p_1/p_2=10$					
$\lambda_1 =$	.1	54.0	56.9	2620.0	1113.
	1.	218.9	80.2	2455.	721.
$\lambda_2 =$	.1	205.9	91.8	2454.	784.
	1.	57.0	45.3	2621.	1050.
$\beta_1 =$	.1	107.8	88.8	5035.	1710.
	1.	148.6	48.8	40.	124.
$\beta_2 =$	.1	124.2	35.1	2368.	980.
	1.	148.7	102.5	2707.	854.

Table F.2 suggests the following comments:

(a) Note the insensitivity of the situation in which $k_1 = k_2 = .1$ to changes in the prior probabilities on target location, as contrasted with the approximate linear effect of such changes when $k_1 = k_2 = 1$. Essentially, this difference results from differences in the peak time relative to the Koopman expected times in the two cases. The peaks occur early in the former case.

(b) Note the decreasing rate of change of measure (d) with increasing p_1/p_2 for the conditions $k_1 = 1.$, $k_2 = .1$.

In contrast to the models of Chapters 2 and 3, we observe that

(a) the use of the Koopman policy will not be adequate in cases in which one has homogeneous visibility conditions,

(b) the error function is not reduced by increasing the available search time (unless extremely large increases are made),

(c) the maximum errors do not occur in situations in which the total available search time is constrained.

(d) the availability of extremely "good" detection devices will not imply that the Koopman scheme will yield small errors.

The above results imply that the decision maker, in general, must obtain accurate estimates of the visibility parameters in order to conduct an effective search, since

the option of increasing the available search time may no longer be cost-effective. The sensitivity analysis of 4.2.3 will give some insight into the sensitivity of the optimal policy and return to these estimates as well as the numerical results just reviewed.[1]

(II) General Single Interval Model

Here the two single-interval models of Chapters 3 and 4 are combined using the limiting values for the state probabilities of an associated alternating renewal process (see Section 4.4.2, $\Pi_i = \frac{\beta_i}{\lambda_i + \beta_i}$).[2] These results were obtained.

(1) The availability of search time on the order of 2-KET (Koopman expected time) will not, in general, reduce the relative error below 5%, in using the SKA. The only exception is the situation in which $p_1/p_2 = 1$ and $k_1 = k_2 = 1$.

(2) In those cases in which the mean time to detect is much less than the mean length of the visibility period (e.g., $k_1 = k_2 = 1$, and $\lambda_1 = \lambda_2 = .1$), the SKA can be used. Under these conditions the probability that the target is visible at the start of the search is at least 0.5 for the levels of β used.

[1] All of the above observations hold for the situation in which the start of the visibility period in the ith region is uniformly distributed on $(0,S_i)$. Section F.4 contains the results of a sensitivity analysis of this situation (Tables F.8 and F.9).

[2] One could, of course, use any prior estimate on Π_i, e.g., one which is independent of λ_i and β_i.

(3) For the short-term search situation, one has the following results:

$p_1/p_2 = 1$

$k_1 \backslash k_2$	0.1	1.0
0.1	Homogeneous visibility conditions yield small errors	See table F.10
1.0	See table F.10	Homogeneous visibility parameters yield small errors[1]

$p_1/p_2 = 2$

$k_1 \backslash k_2$	0.1	1.0
0.1	Homogeneous visibility conditions ($\beta_1 = \beta_2 = .1$) $\rightarrow$ small errors	See table F.11
1.0	See table F.11	Homogeneous visibility conditions with ($\beta_1 = \beta_2 = 1.$) $\rightarrow$ small errors[2]

[1]Recall that for the model of Chapter 3, such conditions implied the optimality of the Koopman policy.

[2]Here one notes that the long-term allocation policies are identical and that the probability of the target being visible at the start of the search is almost 0.5.

Table F.3 summarizes the main effects in terms of measure (d) due to the various search parameters. It suggests the following comments.

(a) For $k_1 = k_2 = .1$, note that the sensitivity to changes in λ_1 and λ_2 increases with increasing p_1/p_2. Changes in the λ's affect the prior probabilities on target visibility at the start of the search. Thus, the general model tends either toward the model of Chapter 3 or Chapter 4 depending upon whether Π can effect the FAR for the general random interval model.

(b) For $k_1 = 1.$ and $k_2 = 0.1$, note the extreme sensitivity to β_1. This causes the general model to tend toward that of Chapter 3 by increasing Π. The reader will recall from our earlier discussions that the model of Cnapter 3 has primarily short-term errors. From approximation policies of Section 4.4.2, observe that for $\beta_1 = 1$, the approximate long-term allocation tends to agree with the Koopman allocation.

To summarize, we have observed that

(a) the Koopman policy can be used effectively in the homogeneous detection, homogeneous visibility condition situation, and

(b) it will also prove adequate in the situations in which the mean time to detect is much less than the mean length of the visible period.

The values of Π, the prior probability vector on target visibility at the start of the search, are of importance here. Since for $\Pi_i \to 1$, the general model reduces to the random

Table F.3

General Random Interval Sensitivity

$p_1/p_2=1$	$k_1 = .1$ $k_2 = .1$	$k_1 = .1$ $k_2 = 1.$	$k_1 = 1.$ $k_2 = .1$	$k_1 = 1.$ $k_2 = 1.$
$\lambda_1 =$ .1	55	205	443	37
1.	51	665	427	59
$\lambda_2 =$ .1	55	443	205	37
1.	51	427	665	59
$\beta_1 =$ .1	60	433	856	60
1.	46	437	14	36
$\beta_2 =$ .1	60	856	433	60
1.	46	14	437	36
$p_1/p_2=2$				
$\lambda_1 =$ .1	35	17	618	101
1.	130	396	782	152
$\lambda_2 =$ .1	105	210	543	99
1.	60	203	857	154
$\beta_1 =$ .1	88	295	1383	185
1.	77	118	17	68
$\beta_2 =$ .1	73	392	792	126
1.	92	21	608	127

interval, initially visible model, and this model has primarily short-term errors under the Koopman policy. If, on the other hand, $\Pi_i \to 0$, one has the random-interval, random-start-time model which has significant long-term errors under the Koopman policy.

(III) Multiple Interval Model

Here it is assumed that the visibility process is characterized by an alternating renewal process with the residency times being exponentially distributed. Again, the probabilities of the target being either visible or masked are taken to be the limiting state probabilities of the associated alternating renewal process. The following results were obtained.

(1) The availability of search time on the order of 2-KET (Koopman expected time) will not, in general, reduce the relative error below 5% in using the SKA.

(2) The SKA can be effectively used in those cases in which the mean time to detect is much less than the mean length of the visible periods.

(3) For the short-term situation, one has the following results:

$p_1/p_2 = 1$

$k_1 \backslash k_2$	0.1	1.0
0.1	Homogeneous visibility conditions → small errors	See table F.12
1.0	See table F.12	Homogeneous visibility conditions → small errors

$p_1/p_2 = 2$

$k_1 \backslash k_2$	0.1	1.0
0.1	Homogeneous visibility conditions → small errors	See table F.13
1.0	See table F.13	Homogeneous visibility conditions with ($\lambda_1 = \lambda_2 = .1$) → small errors

Table F.4 summarizes the main effects in terms of measure (d) due to the various search parameters. It suggests the following comments.

(a) For $k_1 = k_2 = 0.1$, note the increasing sensitivity to λ_1 and β_1 with increasing p_1/p_2 (an increase in p_1 increases the contribution to the optimal return function of the first region).

(b) For $k_1 = 1.$ and $k_2 = 0.1$, note the increasing sensitivity to changes in β_1 (increasing β_1 tends to

Table F.4.

A Multiple Interval Sensitivity

$p_1/p_2=1$		$k_1 = .1$ $k_2 = .1$	$k_1 = .1$ $k_2 = 1.$	$k_1 = 1.$ $k_2 = .1$	$k_1 = 1.$ $k_2 = 1.$
$\lambda_1 =$	.1	51	76	153	36
	1.	93	333	256	91
$\lambda_2 =$	.1	51	153	76	35
	1.	93	256	333	92
$\beta_1 =$	.1	91	334	348	75
	1.	53	75	61	52
$\beta_2 =$	.1	91	348	334	75
	1.	53	61	75	52
$p_1/p_2=2$					
$\lambda_1 =$	.1	43	30	204	33
	1.	109	125	467	190
$\lambda_2 =$	.1	60	28	204	69
	1.	92	127	467	154
$\beta_1 =$	.1	108	129	585	156
	1.	44	26	86	67
$\beta_2 =$	.1	91	111	489	162
	1.	61	44	182	61

yield long-term approximation policies which are close to the Koopman policy).

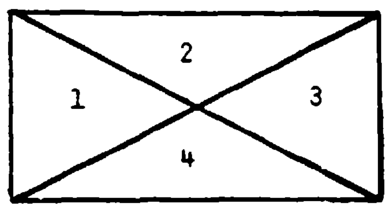
(c) For $k_1 = k_2 = 1$, note the general increase in the sensitivity of all the visibility parameters λ_1, β_1, λ_2, and β_2.

The SKA yields small to moderate errors for homogeneous detection and visibility conditions. Homogeneous visibility conditions will yield, via equation 44 of Section 5.2.1, equal rates (γ_i) for use in the long-term approximate allocations. Thus whenever the detection rates are identical, only short-term errors will occur. On the other hand, heterogeneous visibility conditions yield heterogeneous rates (γ_i) which may differ significantly from the detection rates used by the Koopman policy, resulting in error functions which increase beyond the Koopman expected time. Thus, in certain cases, the decision maker has the full range of options given in Section 2.2.3. For the situation in which the visibility conditions are homogeneous the most cost-effective approach may well be that of increasing the available search effort as opposed to obtaining estimates of the visibility parameters. The data also indicate the sensitivity of the error function to changes in the visibility parameters in the heterogeneous case. Thus, an erroneous assumption on the homogeneity of the visibility conditions could lead to extreme errors if the searcher is

using the Koopman scheme. The sensitivity analysis of Section 5.2.3 and the data of Table F.4 provide insight into the sensitivity of the optimal policy and return to variations in the estimates of the visibility parameters.

F.4 Tables

This section contains the data summarized in the preceding section. For each model the data are arranged in order of increasing ratios of p_1/p_2. The data are arranged in the following order within each cell of the table (for a given ratio of p_1/p_2):



,

where

1 - maximum difference in the probability of detection,

2 - time at which the maximum difference occurs,

3 - percent relative error at the peak difference,

4 - measure (d), defined in Section F.2.

Table F.5

Random Interval Sensitivity Analysis

(Exponential Start Times)

$p_1/p_2=1$			k_1	.1				1.			
			k_2	.1		1.		.1		1.	
λ_1	λ_2	β_2	β_1	.1	1.	.1	1.	.1	1.	.1	1.
.1	.1	.1		10 .023 23 12	5 .06 43 11	13 .13 40 68	4 .02 16 8	13 .126 40 68	5 .09 20 13	3 .02 23 35	2 .136 48 48
		1.		5 .06 43 11	3 .02 21 3	5 .089 20 13	4 .054 13 7	4 .02 16 8	4 .05 13 7	2 .14 48 48	1 .04 31 16
	1.	.1		18 .06 35 32	7 .07 38 13	7 .039 33 30	4 .032 29 15	27 .24 58 205	5 .09 21 14	6 .04 21 63	2 .14 50 50
		1.		2 .013 46 5	6 .04 24 7	4 .026 11 6	3 .013 6 2	33 .27 57 246	5 .07 15 10	2 .07 37 37	2 .05 19 19
1.	.1	.1		18 .06 35 32	2 .01 46 5	27 .24 58 204	33 .269 57 246	7 .04 33 30	4 .03 11 6	6 .04 21 63	2 .07 37 37
		1.		7 .07 38 13	6 .04 24 7	5 .094 21 14	5 .067 15 10	4 .03 29 15	3 .01 6 2	2 .14 50 50	2 .05 19 19
	1.	.1		3 .002 23 4	2 .014 48 5	41 .12 45 241	34 .14 50 222	41 .12 45 241	4 .03 12 6	2 .01 21 21	2 .07 39 39
		1.		2 .014 48 5	1 .004 31 2	4 .03 12 6	3 .014 6 2	34 .14 50 222	3 .014 5 2	2 .07 39 39	1 .02 23 23

Table F.6

Random Interval Sensitivity Analysis

(Exponential Start Times)

$p_1/p_2=2$		k_1	.1	.1	.1	.1	1.	1.	1.	1.
		k_2	.1	.1	1.	1.	.1	.1	1.	1.
α_1	α_2	β_2 \ β_1	.1	1.	.1	1.	.1	1.	.1	1.
.1	.1	.1	16 .035 17 14	11 .02 6 3	7 .05 38 28	3 .05 4 14	20 .23 44 156	5 .07 12 11	7 .06 23 80	3 .14 28 41
		1.	5 .04 45 11	8 .01 2 1	4 .09 33 15	3 .05 21 7	44 .30 41 319	5 .05 8 7	2 .11 56 59	2 .08 21 22
	1.	.1	21 .05 21 23	12 .02 7 5	4 .02 31 14	3 .06 48 10	29 .33 57 292	6 .08 12 12	10 .09 25 126	3 .14 29 43
		1.	2 .01 47 5	9 .01 3 1	3 .03 21 8	2 .02 11 3	33 .34 57 332	5 .06 10 9	2 .06 32 44	2 .09 22 26
1.	.1	.1	17 .05 40 36	23 .07 37 45	40 .16 50 213	1 .01 24 4	13 .08 34 80	5 .02 6 5	4 .02 21 38	2 .06 23 27
		1.	8 .09 74 31	10 .08 51 26	5 .10 35 17	4 .07 24 10	5 .01 13 11	4 .01 3 2	2 .12 57 66	2 .03 17 13
	1.	.1	10 .002 5 3	19 .01 10 10	48 .08 38 194	2 .02 43 9	28 .16 51 252	5 .02 6 5	6 .03 19 60	2 .06 24 28
		1.	3 .013 56 8	7 .02 26 10	4 .04 23 9	2 .02 12 3	34 .18 51 306	4 .01 3 2	2 .06 59 50	2 .04 16 14

Table F.7

Random Interval Sensitivity Analysis
(Exponential Start Times)

$p_1/p_2=10$		k_1	.1				1.			
		k_2	.1		1.		.1		1.	
λ_1	λ_2	β_2 \ β_1	.1	1.	.1	1.	.1	1.	.1	1.
.1	.1	.1	31 .014 3 8	42 .01 2 7	13 .02 8 8	5 .02 8 4	29 .37 48 580	7 .026 3 9	14 .15 24 240	4 .06 8 23
		1.	26 .004 1 2	25 .02 4 7	3 .04 58 16	4 .01 7 3	36 .37 46 690	6 .02 2 5	22 .12 16 251	4 .05 6 17
	1.	.1	35 .02 4 11	– – – –	13 .02 8 9	5 .02 8 4	31 .40 51 659	7 .03 3 9	15 .16 26 279	4 .06 8 23
		1.	41 .015 3 10	24 .003 1 1	2 .02 60 11	4 .01 7 3	31 .40 51 659	7 .02 3 9	18 .14 20 257	4 .05 7 23
	.1	.1	36 .01 11 28	29 .03 28 58	5 .002 6 3	2 .004 8 2	27 .18 43 484	6 .005 1 3	12 .06 20 171	4 .02 4 12
		1.	24 .02 24 41	23 .04 34 56	4 .05 68 25	10 .06 36 33	41 .19 40 683	5 .002 1 1	4 .01 7 63	3 .01 3 7
	1.	.1	26 – 1 1	24 .006 7 12	5 .002 7 3	2 .004 8 2	31 .21 48 620	7 .005 1 4	14 .08 22 220	4 .02 4 12
		1.	23 .004 5 8	23 .008 9 15	2 .025 7 14	2 .003 6 1	33 .21 48 660	6 .002 1 1	20 .07 16 229	3 .01 3 7

Table F.8

Random Interval Sensitivity Analysis
(Uniform Start Times)

$p_1/p_2=1$			k_1	.1				1.			
			k_2	.1		1.		.1		1.	
λ_1	λ_2	S_2	S_1	1	11	1	11	1	11	1	11
.1	.1	1		2 .02 27 3	5 .05 36 9	3 .02 5 2	4 .04 9 5	3 .02 5 2	3 .05 26 10	1 .07 41 20	2 .17 46 46
		11		5 .05 36 9	12 .05 32 19	3 .03 26 10	12 .24 54 84	4 .04 9 5	12 .24 54 84	2 .17 46 46	11 .04 9 50
	1.	1		11 .04 16 8	2 .02 45 5	4 .01 3 2	3 .004 2 1	3 .02 5 2	16 .34 69 140	1 .08 45 23	1 .09 64 32
		11		6 .06 36 12	14 .10 49 35	3 .05 47 18	11 .09 38 54	4 .04 10 5	13 .32 70 118	2 .17 47 47	11 .11 27 148
1.	.1	1		11 .04 16 8	6 .06 36 12	3 .02 5 2	4 .04 10 5	4 .01 3 2	3 .05 47 18	1 .08 45 23	2 .17 47 47
		11		2 .02 45 5	14 .10 49 35	16 .34 68 140	13 .32 70 118	3 .004 2 1	11 .09 38 54	1 .09 64 32	11 .11 27 148
	1.	1		1 .01 41 2	2 .02 46 5	3 .002 1 1	3 .004 2 1	3 .002 1 1	1 .005 28 4	1 .05 35 18	1 .09 65 33
		11		2 .02 46 5	11 .004 9 5	1 .005 28 4	12 .15 62 97	3 .004 2 1	12 .16 62 97	1 .09 65 33	11 .01 5 28

Table F.9

Random Interval Sensitivity Analysis
(Uniform Start Times)

$p_1/p_2=2$			k_1	.1				1.			
			k_2	.1		1.		.1		1.	
λ_1	λ_2	S_2	S_1	1	11	1	11	1	11	1	11
.1	.1	1		9 .01 4 2	4 .05 55 11	2 .03 10 2	3 .06 20 6	4 .01 2 1	22 .37 50 193	2 .07 15 15	2 .15 62 62
		11		11 .02 5 3	13 .06 25 17	3 .06 44 14	11 .10 36 42	5 .02 4 4	12 .37 62 130	2 .12 26 26	11 .17 30 175
	1.	1		3 .004 2 1	2 .01 56 6	2 .01 4 1	3 .01 7 2	4 .01 2 1	15 .42 67 161	2 .08 17 17	1 .08 80 40
		11		12 .02 6 4	14 .07 26 19	3 .07 50 16	5 .02 34 17	5 .03 4 4	13 .41 69 160	2 .12 26 26	11 .21 38 220
1.	.1	1		8 .09 60 24	8 .09 69 29	3 .03 10 3	4 .06 20 9	4 .01 4 3	24 .19 38 160	1 .04 20 10	2 .16 65 65
				19 .10 50 50	14 .06 48 35	2 .02 42 9	12 .20 68 87	4 .001 .3 -	11 .17 53 100	2 .05 15 15	11 .04 12 70
	1.	1		7 .03 31 11	2 .02 73 7	2 .01 4 1	3 .01 7 2	4 .001 .3 -	15 .22 61 160	1 .04 20 10	2 .08 52 52
		11		16 .02 20 17	11 .01 14 8	2 .03 56 12	12 .09 53 67	4 .001 .3 -	12 .21 64 135	2 .05 15 15	11 .085 27 156

Table F. 10

General Random Interval
Sensitivity Analysis

$p_1/p_2=1$			k_1	.1				1.			
			k_2	.1		1.		.1		1.	
λ_1	λ_2	β_2	β_1	.1	1.	.1	1.	.1	1.	.1	1.
.1	.1	.1		- - - -	3 .01 12 4	50 .09 14 91	49 .11 16 105	50 .09 14 91	3 .004 1 -	- - - -	1 .01 3 2
		1.		3 .01 12 4	- - - -	3 .004 1 -	8 .001 - -	49 .11 16 105	8 .001 - -	1 .01 3 2	- - - -
	1.	.1		11 .04 25 14	[illegible] .05 30 9	2 .002 3 1	- - - -	39 .11 24 122	4 .024 5 3	1 .04 27 14	1 .09 31 16
		1.		20 .04 17 17	11 .03 13 7	2 .025 32 8	3 .005 2 -	32 .14 28 117	9 .02 4 5	5 .002 - 1	1 .01 4 2
1.	.1	.1		11 .04 25 14	20 .04 17 17	39 .11 24 122	4 .024 5 3	2 .002 3 1	- - - -	1 .04 27 14	5 .002 - 1
		1.		6 .05 30 9	11 .03 13 7	32 .14 28 117	9 .02 4 5	2 .025 32 8	3 .005 2 -	1 .09 31 16	1 .01 4 2
	1.	.1		- - - -	2 .01 24 2	41 .11 40 214	33 .13 40 198	41 .11 40 244	4 .01 4 2	- - - -	1 .04 26 13
		1.		2 .01 24 2	- - - -	4 .01 4 2	9 .01 4 4	33 .13 46 198	9 .01 4 4	1 .04 26 13	- - - -

Table F.11

General Random Interval
Sensitivity Analysis

$p_1/p_2=2$			k_1	.1				1.			
			k_2	.1		1.		.1		1.	
A_1	A_2	B_2	B_1	.1	1.	.1	1.	.1	1.	.1	1.
.1	.1	.1		- - - -	34 .07 2 3	- - - -	2 .003 2 -	50 .13 17 150	6 .002 - -	6 .002 - 1	1 .013 4 2
		1.		7 .016 9 3	7 .01 4 2	2 .01 2 -	- - - -	47 .14 19 157	10 .001 - -	2 .007 2 2	- - - -
	1.	.1		17 .03 10 10	12 .014 5 3	2 .02 30 6	2 .05 50 10	32 .15 26 147	5 .02 3 3	2 .04 12 12	2 .06 12 12
		1.		27 .03 8 11	20 .01 2 3	- - - -	2 .01 4 1	33 .17 27 157	10 .013 2 4	21 .05 7 70	1 .01 3 2
1.	.1	.1		7 .05 63 23	13 .05 35 24	46 .08 22 107	32 .10 27 92	50 .12 26 230	- - - -	1 .05 43 22	23 .03 4 40
		1.		7 .09 75 28	7 .09 59 22	4 .03 10 5	7 .03 7 6	3 .003 3 2	5 .001 4 4	1 .11 58 29	2 .01 3 3
	1.	.1		- - - -	17 .01 10 10	48 .07 34 174	2 .01 15 3	34 .14 44 260	5 .007 2 2	8 .014 7 28	2 .03 9 9
		1.		7 .02 34 13	7 .02 28 10	3 .01 9 3	10 .01 6 6	34 .16 47 280	10 .007 2 4	1 .04 42 21	- - - -

Table F.12

Multiple Interval
Sensitivity Analysis

$p_1/p_2=1$			k_1	.1	.1	.1	.1	1.	1.	1.	1.
			k_2	.1	.1	1.	1.	.1	.1	1.	1.
λ_1	λ_2	β_2	β_1	.1	1.	.1	1.	.1	1.	.1	1.
.1	.1	.1		– – – –	6 .03 12 4	50 .06 7 46	– – – –	50 .06 7 46	3 .01 3 1	– – – –	1 .01 4 2
.1	.1	1.		6 .03 12 4	– – – –	3 .01 3 1	– – – –	– – – –	– – – –	1 .01 4 2	– – – –
.1	1.	.1		21 .06 22 22	12 .10 29 18	– – – –	3 .03 22 8	18 .10 23 54	5 .04 8 5	1 .04 27 14	1 .09 32 16
.1	1.	1.		– – – –	5 .02 12 3	6 .05 12 9	22 .04 4 12	50 .06 7 46	3 .01 3 1	– – – –	1 .009 3 2
1.	.1	.1		21 .06 22 22	– – – –	18 .10 23 54	50 .06 7 46	– – – –	6 .05 12 9	1 .04 26 13	– – – –
1.	.1	1.		12 .10 29 18	5 .02 13 3	5 .04 8 5	3 .01 3 1	3 .03 23 8	22 .04 4 12	1 .09 32 16	1 .009 3 2
1.	1.	.1		– – – –	19 .08 25 25	33 .18 45 194	– – – –	33 .18 45 194	8 .12 25 25	– – – –	3 .07 20 30
1.	1.	1.		19 .08 25 25	– – – –	8 .12 25 25	5 .05 12 8	– – – –	5 .05 12 8	3 .07 20 30	– – – –

Table F.13

Multiple Interval
Sensitivity Analysis

$p_1/p_2=2$		k_1	.1				1.			
		k_2	.1		1.		.1		1.	
λ_1	λ_2	β_2 \ δ_1	.1	1.	.1	1.	.1	1.	.1	1.
.1	.1	.1	.01; 20; 2; 2	.02; 13; 5; 3	.001; 3; 1; 0	.004; 2; 3; 0	.07; 20; 12; 41	.01; 4; 2; 1	-; -; -; -	.02; 1; 4; 3
.1	.1	1.	.014; 7; 7; 3	-; -; -; -	.01; 3; 4; 1	-; -; -; -	.05; 20; 8; 26	.003; 4; 4; -	.02; 1; 8; 3	-; -; -; -
.1	1.	.1	.07; 20; 18; 18	.07; 20; 13; 13	.018; 2; 37; 6	.06; 2; 65; 11	.16; 20; 27; 95	.03; 6; 5; 5	.04; 2; 11; 11	.07; 2; 13; 13
.1	1.	1.	.003; 12; 1; 1	.02; 12; 4; 3	.03; 8; 7; 7	.02; 20; 2; 5	.06; 20; 10; 35	.01; 4; 2; 1	-; -; -; -	.02; 1; 4; 3
1.	.1	.1	.05; 7; 56; 20	.02; 20; 4; 11	.05; 10; 19; 20	-; -; -; -	.12; 20; 29; 102	.08; 7; 13; 16	.05; 1; 54; 22	.03; 5; 4; 11
1.	.1	1.	.12; 7; 75; 27	.006; 7; 3; 1	.05; 4; 14; 6	.01; 3; 4; 1	-; -; -; -	.05; 16; 6; 18	.12; 1; 70; 28	.01; 1; 6; 2
1.	1.	.1	.01; 20; 13; 13	.08; 20; 18; 18	.07; 20; 32; 68	.02; 2; 38; 6	.24; 20; 56; 193	.13; 9; 20; 31	.03; 14; 10; 70	.09; 4; 18; 32
1.	1.	1.	.06; 10; 48; 24	.01; 14; 3; 2	.09; 7; 30; 21	.02; 4; 9; 3	.11; 20; 15; 88	.08; 7; 13; 15	.06; 1; 56; 22	.01; 3; 2; 3

Appendix G

SOME RESULTS FROM THE THEORY OF SEMI-MARKOV PROCESSES

In this appendix, we list some theorems on Semi-Markov processes due to Pyke (1961). First some definitions, let $G_{ij}(t)$ denote the distribution function for the first passage time, i.e.,

$$G_{ij}(t) = \Pr[N_j(t) > 0 | X_0 = i] \equiv \Pr\{t_{j_1} \leq t | X_0 = i\} ,$$

where t_{j_1} denotes the time of the first passage to state j and $N_j(t)$ denotes the number of entries to state j in time t. Let $P_{ij}(t)$ be defined as

$$P_{ij}(t) = \Pr\{X_t = j | X_0 = i\} .$$

The Laplace transforms of these are, respectively,

$$g_{ij}(s) = \int_0^\infty e^{-st} \, dG_{ij}(t) ,$$

and

$$\Pi_{ij}(s) = \int_0^\infty e^{-st} \, dP_{ij}(t) .$$

The Laplace transform of Q, the associated semi-Markov matrix, is denoted by

$$q = \{q_{ij}(s)\} = \int_0^\infty e^{-st} \, dQ_{ij}(t) .$$

Pyke (1961) proves the following theorems relating P and G.

Lemma 1.0

For all $t \geq 0$, $s > 0$

$$P_{ij} = \delta_{ij} - \sum_{K=1}^{m} [\delta_{ij} - P_{Kj}(t)]*Q_{iK}(t)$$

$$\Pi_{ij}(s) = \delta_{ij} - \sum_{K=1}^{m} [\delta_{ij} - \Pi_{Kj}(s)]q_{iK}(s) .$$

Lemma 2.0

For $t \geq 0$, $s > 0$,

$$G_{ij}(t) = \sum_{K=1}^{m} G_{Kj}(t)*Q_{iK}(t) + [1 - G_{jj}(t)]*Q_{ij}(t)$$

$$g_{ij}(s) = \sum_{K=1}^{m} g_{Kj}(s)q_{iK}(s) + [1 - g_{jj}(s)]q_{ij}(s) .$$

Theorem 1

For $t \geq 0$, $s > 0$,

$$P_{ij}(t) = P_{jj}(t)*G_{ij}(t) + \delta_{ij}[1 - \Lambda_i(t)]$$

and

$$\Pi_{ij}(s) = \Pi_{jj}(s) g_{ij}(s), \; i \neq j$$

$$\Pi_{jj}(s) = \frac{1 - h_j(s)}{1 - g_{jj}(s)}$$

where

$$\Lambda_i(t) = \sum_{j=1}^{M} Q_{ij}(t), \quad h_{ij}(s) = \int_0^\infty e^{-st} d\Lambda_j(t) \; .$$

Next we define the convolution on matrix-valued functions as

$$(K*L)_{ij} = \sum_{K=1}^{m} K_{iK} * L_{Kj} \; ,$$

and

$$K^o = I, \; K^{(N)} = K^{(N-1)} * K, \; (I - K)^{(-1)} = \sum_{n=0}^{\infty} K^{(N)}.$$

Theorem 2

Given $P = \{P_{ij}\}$, $Q = \{Q_{ij}\}$, $H = \{\delta_{ij} \Lambda_i\}$ and their L - S transforms Π, q, h, then the following relationship holds

$$P = (I - Q)^{-1} * (I - \Lambda), \quad \Pi = (I - q)^{-1} (I - h) \; .$$

For any square matrix (or matrix valued function) $A = (a_{ij})$, define the diagonal and off-diagonal parts of A by

$$dA = (\delta_{ij}A_{ij})$$

$$_{o}A = A - dA .$$

Theorem 3

As defined on $s \in (0,\infty)$,

$$g = q\Pi(d\Pi)^{-1}$$

As a result of this theorem, one can obtain the mean recurrence times $u = (u_{ij})$ from

$$u = \lim_{s \to 0} \frac{1}{s} [\, \mathbb{1} - g \,] ,$$

where $\mathbb{1}$ is an nxn matrix each entry being unity.

BIBLIOGRAPHY

Bellman, R.E., and Dreyfus, S.E. *Applied Dynamic Programming*. Princeton University Press, Princeton, New Jersey (1962).

Black, William Lawrence. "Discrete Sequential Search." *Information and Control*, 8 (1965), 152-62.

Blackman, Nelson. "Prolegomena to Optimum Discrete Search Procedures." *Naval Research Logistics Quarterly*, 6, 4 (1959), 273-81.

________, and Proschan, Frank. "Optimum Search for Objects Having Unknown Arrival Times." *Operations Research*, 7, 5 (1959), 625-38.

Bonder, Seth. "Needs in Operations Research Education." IE Technical Report 70-10, Department of Industrial Engineering, College of Engineering, The University of Michigan, 22 April 1970.

________. Personal communciation with A. Booth of the British Defense Operational Analysis Establishment, Quadripartite Meeting (1970).

________. "Preliminary Modeling of Reconnaissance in Force." SRL 1957 WP 68-10, Systems Research Laboratory, The University of Michigan, 4 November 1968.

________, and Farrell, Robert. "Development of Models for Defe[illegible]tems Planning." SRL 2147 TR 70-2, Systems [illegible] Laboratory, The University of Michigan, Se[illegible] 1970.

Brand, L. *Ad[illegible] Calculus*, Wiley and Sons, New York, 1960.

Charnes, A., and Cooper, W.W. "The Theory of Search: Optimum Distribution of Search Effort." *Management Science*, 5 (1958), 44-50.

________, and Schroder, R. "On Some Stochastic Tactical Anti-Submarine Games." *Naval Research Logistics Quarterly*, 14, 3 (1967), 291-311.

Chew, M., Jr. "A Sequential Search Procedure." *Annals of Mathematical Statistics*, 38 (1967), 494-502.

Gilbert, E.N. "Optimal Search Strategies." *SIAM Journal*, 7 (1959), 413-24.

Guenin, Jacques, de. "Optimum Distribution of Effort: An Extension of the Koopman Basic Theory." *Operations Research*, 9, 1 (1961), 1-7.

Disney, R. "A Probability Structure for the Intervisibility Process." SRL 1957 WP 68-11, Systems Research Laboratory, The University of Michigan, 8 November 1968.

Dobbie, J.M. "Search Theory: A Sequential Approach." *Naval Research Logistics Quarterly*, 10 (1963), 313-34.

Howard, R.A. *Dynamic Programming and Markov Processes*. M.I.T. Press, Cambridge, Massachusetts, 1960.

Kadane, J. "Discrete Search and the Neyman-Pearson Lemma." *Journal of Mathematical Analysis and Application*, 22 (1968), 156-71.

Klein, M. "A Note on Sequential Search." *Naval Research Logistics Quarterly*, 15, 3 (1968), 469-74.

Koopman, Bernard Osgood. "Search and Screening." OEG Report 56 (ATI 64 627), Operations Evaluation Group, Office of the Chief of Naval Operations, Washington, D.C., 1946.

________________________. "Hide and Seek Games." *NATO Conference Proceedings*, 1963, 40-56.

Kronz, R.L. "Effects of the Response Process in Search Models with False Detections." SRL 2147 TR 71-2, Systems Research Laboratory, The University of Michigan (November 1971).

MacQueen, J., and Miller, R.G., Jr. "Optimal Persistence Policies." *Operations Research*, 8, 3 (May-June 1960), 362-80.

Matula, David. "A Periodic Optimal Search." *American Mathematical Monthly*, 71 (1964), 15-21.

Moore, Michael L. "A Review of Search and Reconnaissance Theory Literature." SRL 2147 TR 70-1, Systems Research Laboratory, The University of Michigan, September 1970.

Morris, W.T. "On the Art of Modeling." *Management Science*, 13, B-707-17 (August 1967).

Neuts, Marcel F. "A Multistage Search Game." *SIAM Review*, 11 (1963), 502-07.

Norris, R.C. "Studies in Search for a Conscious Evader." Technical Report 279, Lincoln Laboratory, Massachusetts Institute of Technology, 14 September 1962.

Onaga, K. "Optimal Search for Detecting a Hidden Object." *SIAM Journal of Applied Mathematics*, 20, 2 (March 1971).

Parzen, E.M. *Stochastic Processes*. Holden-Day, San Francisco (1962).

Pollock, Stephen M. "Optimal Sequential Strategies for Two Region Search when Effort Is Quantized." Interim Technical Report 14, Operations Research Center, Massachusetts Institute of Technology (May 1960).

________________. "Sequential Search and Detection." Technical Report 5, Operations Research Center, Massachusetts Institute of Technology (May 1964).

________________. "A Simple Model of Search for a Moving Target." *Operations Research*, 18, 5 (1970), 883-903.

Pontrayagin, L.S., et al. *The Mathematical Theory of Optimal Processes*. Interscience, New York (1962).

Pyke, R. "Markov Renewal Processes with Finitely Many States." *Annals of Mathematical Statistics*, 32 (1961), 1243-59.

Stollmack, S. "The Tank Weapon System." RF-573, AR 68-1, Systems Research Group, Department of Industrial Engineering, The Ohio State University (September 1968), Chapters 2-5.

Stone, L.D. "Total Optimality of Incrementally Optimal Allocations." Daniel H. Wagner, Associates, Paoli, Pennsylvania (January 11, 1971).

Vector Research, Inc. "Modification and Improvement of Differential Models of Combat, I and II." VRI-2 FR 70-1, Ann Arbor, Michigan (October 1970).

Wagner, D.H. "Nonlinear Functional Version of the Neyman-Pearson Lemma." *SIAM Review*, 11, 1 (January 1969), 52-65.

Wagner, Harvey. "The ABC's of OR." *Operations Research*, 19, 6 (1971), 1259-1281.

Wagner, Harvey. *Principles of Operations Research*. Prentice-Hall, Englewood Cliffs, New Jersey (1970).

Zahl, Samuel. "An Allocation Problem with Applications to Operations Research and Statistics." *Operations Research*, 11, 3 (1963), 426-41.

FOREWORD

Under Contract DAHC-04-69-C-0056 between IIT Research Institute (IITRI) and the U. S. Army Research Office-Durham, IITRI is performing a study of fragmentation hazards to unprotected personnel. This is the final report of the study which was carried out during the period July 1971 to December 1971.

The work was conducted for the Department of Defense Explosives Safety Board and supervised by Col. W. Cameron III, Chairman, Lt. Col. J. D. Coder, Project Manager, Dr. T. A. Zaker, Explosives Scientist and Mr. R. G. Perkins, Safety Engineer. Principal contributors to the presently reported work include L. A. C. Barbarek, D. I. Feinstein, H. H. Nagaoka, J. D. Rouse and J. R. Wingfield. Special acknowledgement is made to Dr. Zaker for providing trajectory algorithms and overall technical direction to this study.

Respectfully submitted,
IIT RESEARCH INSTITUTE

D. I. Feinstein
Research Engineer

APPROVED:

E. P. Bergmann, Manager
Civil Engineering Systems
and Explosives

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) IIT Research Institute 10 West 35th Street Chicago, Illinois 60616	2a. REPORT SECURITY CLASSIFICATION 2b. GROUP N/A

3. REPORT TITLE
FRAGMENTATION HAZARDS TO UNPROTECTED PERSONNEL

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Final Report (July 1971 to December 1971)

5. AUTHOR(S) (First name, middle initial, last name)
D. I. Feinstein

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
January 1972	202	5

8a. CONTRACT OR GRANT NO. DAHC-04-69-C-0056 b. PROJECT NO c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) J-6176 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Dept.of Defense Explosives Safety Bd. Forrestal Building, Room GB270 Washington, D. C. 20314

13. ABSTRACT A computer program has been developed and documented that generates the information necessary to establish minimum separation distances between various munition types and personnel in order to mitigate fragment hazards. This information includes the fragment density at any point from an accidental detonation and the probability of injury to personnel within the hazardous area.

Published munition effectiveness data have been modified, utilizing statistical techniques, to emphasize a conservative approach, with respect to safety, by a better resolution of heavier fragments. The modified munition effectiveness data, which is input to the computer program have been included as a technical data appendix.

Computed results have been obtained for seven munitions and are presented in the form of contours of constant fragment density and damage probability in the horizontal plane surrounding the munition of interest. In addition, these contours have been simplified to yield an average value of the hazard as a function of range from the accidental explosion.

DD FORM 1 NOV 65 1473

UNCLASSIFIED
Security Classification

ABSTRACT

A computer program has been developed and documented that generates the information necessary to establish minimum separation distances between various munition types and personnel in order to mitigate fragment hazards. This information includes the fragment density at any point from an accidental detonation and the probability of injury to personnel within the hazardous area.

Published munition effectiveness data have been modified, utilizing statistical techniques, to emphasize a conservative approach, with respect to safety, by a better resolution of heavier fragments. The modified munition effectiveness data, which is input to the computer program, have been included as a technical data appendix.

Computed results have been obtained for seven munitions and are presented in the form of contours of constant fragment density and damage probability in the horizontal plane surrounding the munition of interest. In addition, these contours have been simplified to yield an average value of the hazard as a function of range from the accidental explosion.

CONTENTS

ILLUSTRATIONS

TABLES

FRAGMENTATION HAZARDS TO UNPROTECTED PERSONNEL

1. INTRODUCTION

Existing quantity-distance standards for the manufacture, handling, and storage of munitions are based on the net weight of explosive filler contained in the devices in an unsubdivided magazine, or operating building unit. Safe distances are prescribed in tabular form essentially proportional to the cube root of the explosive weight.

Because peak blast pressures from explosions of different yield are the same at distances scaled by the cube root of the respective explosive weights, existing standards imply that the acceptable risk to a given target is based on a peak blast overpressure criterion alone. On the other hand, the field of fragments projected to the far field from accidental explosion of a munition store, consisting of inert munition component fragments and secondary fragments from any enclosure, does not satisfy the same similarity rules as does the airblast. Thus, defining an acceptable blast overpressure level at a target implies the acceptance of different levels of risk of damage by fragments, depending on the quantity and composition of the munition store and its enclosure.

To develop quantity-distance standards based on consistent blast and fragment hazard levels requires determination of the damage risk due to fragments from accidental explosions as a function of the quantity and type of munitions, and of the characteristics of the source environment and the vulnerability of the target. A previous report (Ref. 1),* was a first attempt aimed at applying engineering analysis, supplementary experimental efforts, and currently available data on fragmentation and damage criteria to the problem of estimating fragment hazards at explosives manufacturing and storage sites. While that report resulted in a fragment hazard model and a set of fragment density and damage probability maps for a variety of weapons and targets, this report is concerned with:

*References listed at the end of the text.

(1) Refining that model and making its corresponding computer algorithm usable by personnel moderately experienced at ballistic calculations,

(2) Revising near-field fragment mass distributions to more accurately reflect their use as input in estimating fragment hazards,

(3) Developing a new set of fragment density and damage probability maps; studying the effect of specific personnel protection criteria for unprotected personnel, and

(4) Developing simplified two-dimensional relationships of fragment density and damage probability as a function of range.

1.1 Problem Background

Under Contract No. DAHC-04-69-C-0056 with the U.S. Army Research Office-Durham, IITRI has been conducting a series of investigations concerning fragment hazards associated with accidental detonation of munitions. This work has been performed under the direction of the Department of Defense Explosive Safety Board.

Phase I of this study was concerned with establishing quantitative damage criteria in terms of fragment mass, velocity, and attack angle for various targets including standing personnel, vehicles, aircraft, buildings and open weapon stores. In Phase II an analytical model was developed to predict the density of fragments and the probability of damage to the targets considered in Phase I from explosion of individual munitions of various types. These included gun projectiles and general-purpose bombs. Here damage probability contours were obtained in polar coordinates for a horizontal orientation of the munition axis in each case. Phase III attempted to extend the fragment hazard model for individual munitions to the case of multiple munitions in open stores (Ref. 2). The result was a limited demonstration that an analytic model could be developed to describe the initial fragment field of a stack of munitions. However, it was also brought out that this initial fragment field was often related to munition case design, stack configuration and mode of initiation.

1.2 Program Objectives

In the current research activity the intent has been to develop an analytic tool, usable by personnel moderately experienced at ballistic calculations, and capable of generating the information necessary to establish the minimum separation distance to personnel. The objectives of the study were:

(1) To develop and document a computer algorithm which uses munition effectiveness tables as input and computes fragment densities and damage probabilities to unprotected personnel based upon the following criteria:

- A hazardous fragment has a kinetic energy of 58 ft-lbs or greater, and
- An acceptable density of hazardous fragments is not more than one per 600 sq ft.

(2) To develop a rational scheme for revising published munition effectiveness data to more accurately reflect its use as input in estimating fragment hazards.

(3) To develop a simplified means of relating fragment density and damage probability to a radial distance over a fixed sector of the ground plane.

(4) To utilize the computer algorithm and the revised data in order to compute the fragment density and damage probability from the explosion of a single round of each of the following seven munitions:

- 500 lb low drag bomb Mark 82 Mod 1 (H-6 load)
- 750 lb Bomb M117A2 (Tritional Load)
- 105mm Howitzer Shell M1 (Composition B Load)
- 155mm Howitzer Shell M107 (Composition B Load)
- 175mm Gun Shell M437A2 (Composition B Load)
- 5"/38 Projectile Mark 49 (COMP A-3 Load)
- 8"/55 Projectile Mark 25 (Explosive D Load)

1.3 Program Accomplishments

The major result of this study has been the development and documentation of a computer algorithm which will allow safety personnel, moderately experienced at ballistic calculations, to make the computations necessary to establish separation distances for personnel due to fragment hazards. A rational statistical scheme has also been developed and demonstrated which revises published munition effectiveness data to more conservatively reflect its use in estimating fragment hazards. A set of computations were made for seven munitions and the results are presented as contour maps of total fragment number densities, damaging fragment number densities and injury probabilities. The contour maps have also been simplified to yield curves of number density and injury probability as a function of radial distance from the munition source within a constant sector of the ground plane. (i.e., the nose, base and side-spray sectors) These results serve both to demonstrate the capability of the computer algorithm and also as design aids for explosive safety personnel.

1.4 Program Highlights

The following sections of this report are organized in such a way as to first present the analysis on which the fragment hazard computer model is based, to next present an analysis of the munition effectiveness data which is input to that computer model and to finally summarize the study in the form of conclusions reached and recommendations as to model improvement.

Appendices, covering the study's deliverable items, are included following the conclusion of the report. These appendices include a user manual for operation of the fragment hazard computer program, a listing of the computer program, the contours and simplified curves describing the fragment hazard associated with unprotected personnel exposed to these

seven munitions. The revised munition effectiveness data for the seven munitions were published as a separate classified document.

2. THE FRAGMENT HAZARD MODEL

This section describes the analysis which forms the basis for the computational model for predicting fragment density and injury probability contours for various munitions The mathematical model, illustrated in Fig. 1, is limited to the consideration of the single munition without environmenta protection. The model has been programmed for digital computa tion, is modular and accepts as inputs:

(1) The spatial distribution of fragment masses and velocities for single munitions, which are defined for intervals of polar angle. (i.e., munition effectiveness tables)

(2) The k-factors for the individual munitions, which express the relationship between fragment masses and projected areas for various munition types.

(3) Vulnerability criteria for targets of interest. (e.g., kinetic energy and critical densities)

Output of the model includes:

(1) Fragment density contours showing distances to isodensity lines for all azimuths. Contours can be printed for all fragments or for various classes of fragments.

(2) Injury/damage probability contours, showing ground distances to isoprobability curves at all azimuths for various munition/target combinations.

Elements of this model include the following steps:

(1) Munition performance data is converted to an internal form. All data is normalized into a a common internal form, wherein the average fragment velocity and the average mass and number density for each of several fragment mass categories are represented as tabular functions defined everywhere on the surface of a sphere of unit radius located at the origin. Gaps in the original data are filled in, and in addition the data may be smoothed if this seems desirable.

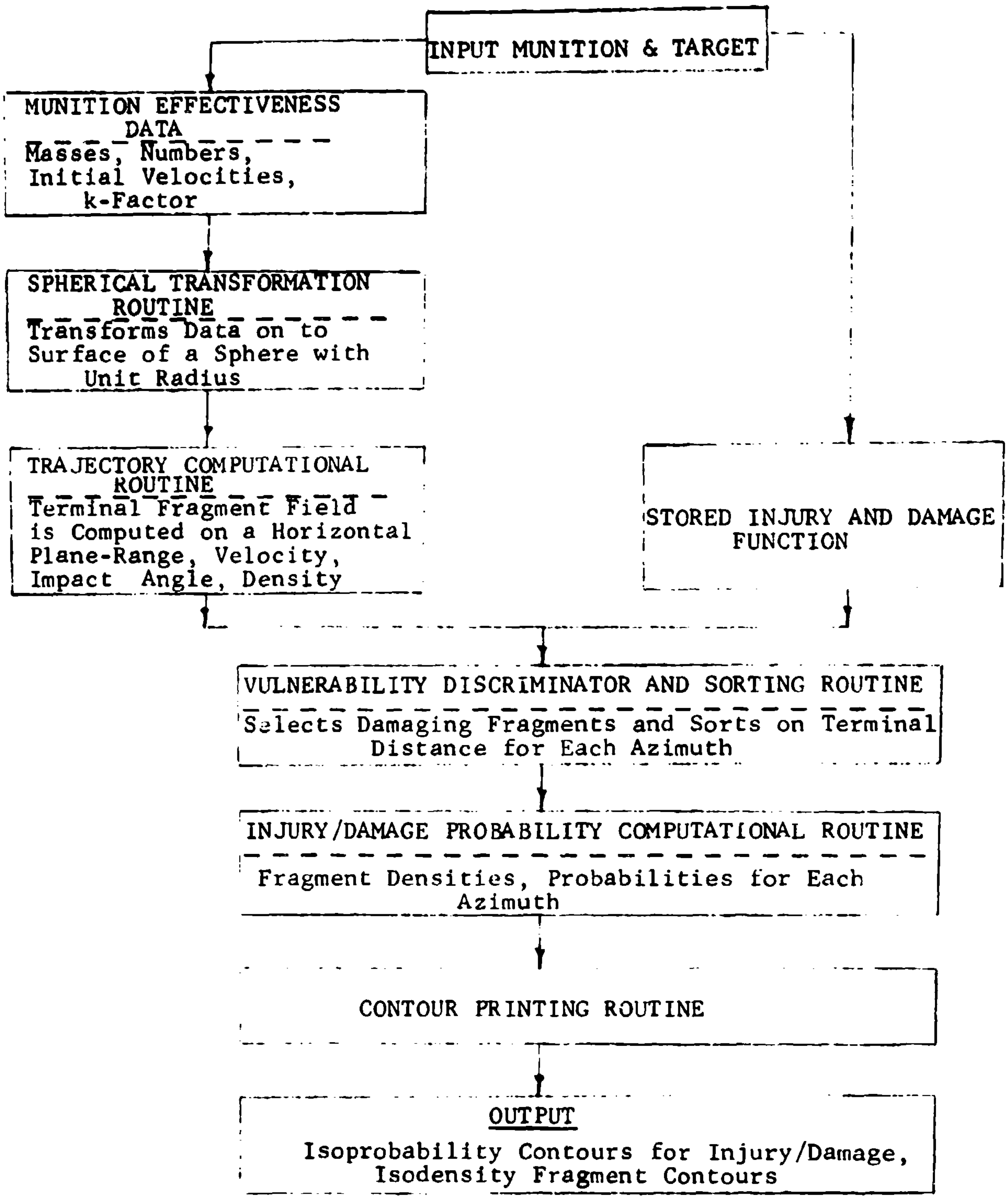


Fig. 1 FLOW CHART FRAGMENT HAZARD MODEL

(2) A terminal fragment field is computed on a selected horizontal plane (i.e., the plane is above, below or passing through the origin) utilizing routines for evaluating fragment trajectories. The quantities computed for each mass category are impact velocity, impact angle and number of fragments per sq ft in a plane normal to the impact angle. These quantities are expressed as tabular functions of range and azimuth for output processing.

(3) Selected functions of the fragment field quantities computed in step 2 are computed and plotted in this step. A wide range of functions are available. The basic functions involve the fragment field alone, and do not consider characteristics of a target. Examples of these functions are number of fragments per sq ft, and total fragment kinetic energy per sq ft.

The target functions use target characteristics to determine the number of damaging fragments and the probability of damage at every point in the field. These functions use tables or formulas to determine whether or not a fragment is damaging, by finding the minimum velocity required by that fragment to damage the target. The fragment is considered to be damaging if its velocity exceeds the threshold.

(4) Fragment field functions may be plotted either on the printer, or on an off-line plotter if the output tape is run through an appropriate post-processor.

2.1 Trajectory Analysis

Large quantities of terminal ballistic property data are used in developing the outputs of the computational model described above. These data are generated from the equations of motion for the fragments. Since these computations represent the bulk of the computational burden involved in exercising the model and their accurate evaluation is essential, it is desirable to utilize a highly efficient numerical procedure for calculating trajectories.

Figure 2 illustrates the motion of a fragment moving under the influence of aerodynamic drag and gravity forces in nonrotating local coordinates $\bar{x}$, $\bar{y}$ tangent and normal to the trajectory. The corresponding equations of motion are:

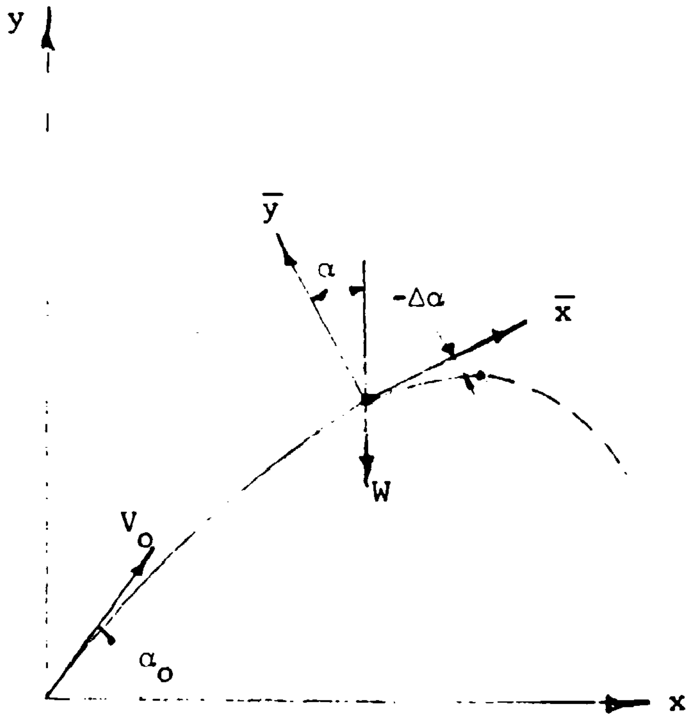


Fig. 2 COORDINATE SYSTEMS AND TRAJECTORY GEOMETRY

$$\ddot{\bar{x}} + \beta v \dot{\bar{x}} + g \sin\alpha = 0 \qquad (1)$$

$$\ddot{\bar{y}} + \beta v \dot{\bar{y}} + g \cos\alpha = 0 \qquad (2)$$

where dots denote differentiation with respect to time t. In these equations, g is the acceleration of gravity, v is the speed in the path, and α is the angle between the $\bar{x}$ - axis and the horizontal. Instantaneously we have $\dot{\bar{x}} = v$ and $\dot{\bar{y}} = 0$.

The aerodynamic coefficient β is given by

$$\beta = C_D w\, A/2W \qquad (3)$$

where C_D is the drag coefficient, w is the specific weight of air, A is the cross-sectional area of the fragment normal to the flight direction, and W is the fragment weight. The fragment area and weight are related empirically (Ref. 3) through a ballistic density k as follows

$$W = k\, A^{3/2} \qquad (4)$$

In terms of k, the aerodynamic coefficient becomes

$$\beta = C_D\, w/2(k^2 W)^{1/3} \qquad (5)$$

An approximate local solution to the equations of motion is obtained by separating the displacement into two parts, one a basic solution satisfying the local initial conditions and the equations of motion with gravity absent, and the other a pair of perturbations satisfying the linearized residual equations (Ref. 4). The results, applicable for small departures of the trajectory from the local initial tangent, are equivalent to difference equations appropriate to an arbitrary time step in a numerical integration of the complete trajectory.

The displacement $\bar{x}$ is assumed to be of the form

$$\bar{x} = \bar{x}_o + \bar{x}_p \qquad (6)$$

where the basic solution $\bar{x}_o$ satisfies

$$\ddot{\bar{x}}_o + \beta\dot{\bar{x}}_o^2 = 0 \tag{7}$$

and the initial condition $\dot{\bar{x}}_o = v$, while the perturbation $\bar{x}_p$ satisfies the associated residual of Eq. (1).

The drag coefficient C_D in general depends on the Mach number, and the atmospheric weight density w is a function of altitude. If both of these factors are assumed to be constant during the time interval of interest, however, the aerodynamic coefficient β is a constant and Eq. (7) is easily integrated. The results are

$$\bar{x}_o = [\log (1+u)] / \beta \tag{8}$$

$$\dot{\bar{x}}_o = v_o/(1+u) \tag{9}$$

where

$$u \equiv \beta v_o t \tag{10}$$

In these expressions, t is measured from the time at which the fragment is at the local coordinate origin in Fig. 1, and v_o is the value of v at that time.

Substituting Eq. (6) and (7) into the equations of motion, expanding v in binomial series, and neglecting terms of second order and higher in $\dot{\bar{x}}_p$ and $\dot{\bar{y}}$, we reach the following results:

$$\ddot{\bar{x}}_p + 2\beta\dot{\bar{x}}_o\dot{\bar{x}}_p + g \sin\alpha = 0 \tag{11}$$

$$\ddot{\bar{y}} + \beta\dot{\bar{x}}_o\dot{\bar{y}} + g \cos\alpha = 0 \tag{12}$$

These equations are linear in the displacement perturbations $\bar{x}_p$ and $\bar{y}$, and can be integrated analytically by standard methods. The displacement and velocity perturbations are

$$\bar{x}_p = -(g/2)\ t^2 \sin\alpha\ (1+u/3)/(1+u) \tag{13}$$

$$\bar{y} = -(g/2)\ t^2 \cos\alpha\,[\,u(1+u/2)-\log(1+u)]/u^2 \tag{14}$$

$$\dot{\bar{x}}_p = -gt \sin\alpha\,[\,1+u(1+u/3)\,]/(1+u)^2 \tag{15}$$

$$\dot{\bar{y}} = -gt \cos\alpha\ (1+u/2)/(1+u) \tag{16}$$

where u is defined as before by Eq. (10).

The leading factors on the right in the foregoing equations express the position and velocity changes due to gravity in the elementary case of a drag-free trajectory. The multipliers containing u all approach unity as u vanishes, and can be viewed as corrections on the effect of gravity due to drag.

The drag coefficient and atmospheric density are assumed to be constant during each time step at their values at the beginning of the step. The method is self-starting in that the position and velocity changes are computed from initial values at the current step only.

Initial values $v = V_o$ and $\alpha = \alpha_o$ are assumed to be given at the fixed coordinate origin in Fig. 2. Equations (8), (9), and (13) through (16) give directly the displacement and velocity components after a typical time step t in the local coordinates. With respect to the fixed coordinates, the displacements during the time step are obtained from the relations

$$\Delta x = \bar{x} \cos\alpha - \bar{y} \sin\alpha \tag{17}$$

$$\Delta y = \bar{x} \sin\alpha + \bar{y} \cos\alpha \tag{18}$$

while the rotation of the trajectory tangent is given by

$$\Delta\alpha = \tan^{-1} (\dot{\bar{y}}/\dot{\bar{x}}) \tag{19}$$

For low register trajectories, (i.e., those launched at angles less than that corresponding to maximum range) the above analytic perturbation equations with $\alpha = \alpha_o$ furnish an approximate solution for the complete trajectory in one time step t, the total time of flight. At impact, the expressions for the position coordinates x and y are of the form:

$$x = (\bar{x}_o + \bar{x}_p) \cos\alpha_o - \bar{y} \sin\alpha_o \tag{20}$$

$$y = (\bar{x}_o + \bar{x}_p) \sin\alpha_o + \bar{y} \cos\alpha_o = 0 \tag{21}$$

where $\bar{x}_o$, $\bar{x}_p$, and $\bar{y}$ are given by Eq. (8), (13), and (14).

2.2 Fragment Density Computation

Munition effectiveness data are used to derive initial conditions for the ballistic trajectories of fragments. These data are in the form of initial velocity and number of fragments, in each of several mass categories, and are functions of polar angle measured from the nose of the munition. The resulting fragment density at any point of interest can be computed deterministically from the known terminal points of fragments in all the mass intervals in each polar zone.

An individual munition is regarded as a nonisotropic point source of fragments which is rotationally symmetric about its longitudinal (nose-to-base) axis. Thus, the properties of the fragments emitted by a single munition are functions of polar angle θ measured from the nose. The format of typical munition effectiveness data (Ref. 5) is shown in Table 1. Fragments in all mass intervals are assumed to be emitted from a given polar zone at the same velocity.

A single munition is assumed to be detonated at a level ground surface, with the munition axis horizontal. In order to determine the probability of damage to targets at various distances and directions from the source, it is first necessary to calculate the number densities of fragments of different terminal ballistic properties in the field surrounding the source.

The fragments from a munition, considered as a point source on the ground, may be regarded as originating from positions on a hemispherical envelope enclosing the source. Each point of origin on the hemisphere defines an elevation angle α_o and an azimuth ϕ in the horizontal plane measured from the nose of the munition.

In Fig. 3, let u, v, and w be rectangular coordinate axes centered at the source, with w the vertical direction and the nose of the munition in the positive v-direction. Let R_o be the (arbitrary) hemisphere radius, and θ be the polar angle measured from the nose of the munition.

TABLE 1

TYPICAL FORMAT OF EXPERIMENTAL DATA FOR MUNITION OF INTEREST

Mass Intervals (grains) / Polar Zone (degrees)	o - m_1		m_1 - m_2		. . . .	m_8 - m_9		Above m_9		Total Number of Fragments	Initial Velocity
	W_{AV}	$\overline{N}$	W_{AV}	$\overline{N}$		W_{AV}	$\overline{N}$	W_{AV}	$\overline{N}$		
0 - 5	-	-	-	-	. . . .	-	-	-	-	-	-
5 - 10	-	-	-	-	. . . .	-	-	-	-	-	-
10 - 15	-	-	-	-	. . . .	-	-	-	-	-	-
.	.	.	.	.	. . . .	.	.	.	.	.	.
.	.	.	.	.	. . . .	.	.	.	.	.	.
.	.	.	.	.	. . . .	.	.	.	.	.	.
.	.	.	.	.	. . . .	.	.	.	.	.	.
170 - 175	-	-	-	-	. . . .	-	-	-	-	-	-
175 - 180	-	-	-	-	. . . .	-	-	-	-	-	-

Where: W_{AV} is average weight, $\overline{N}$ is average number of fragments.

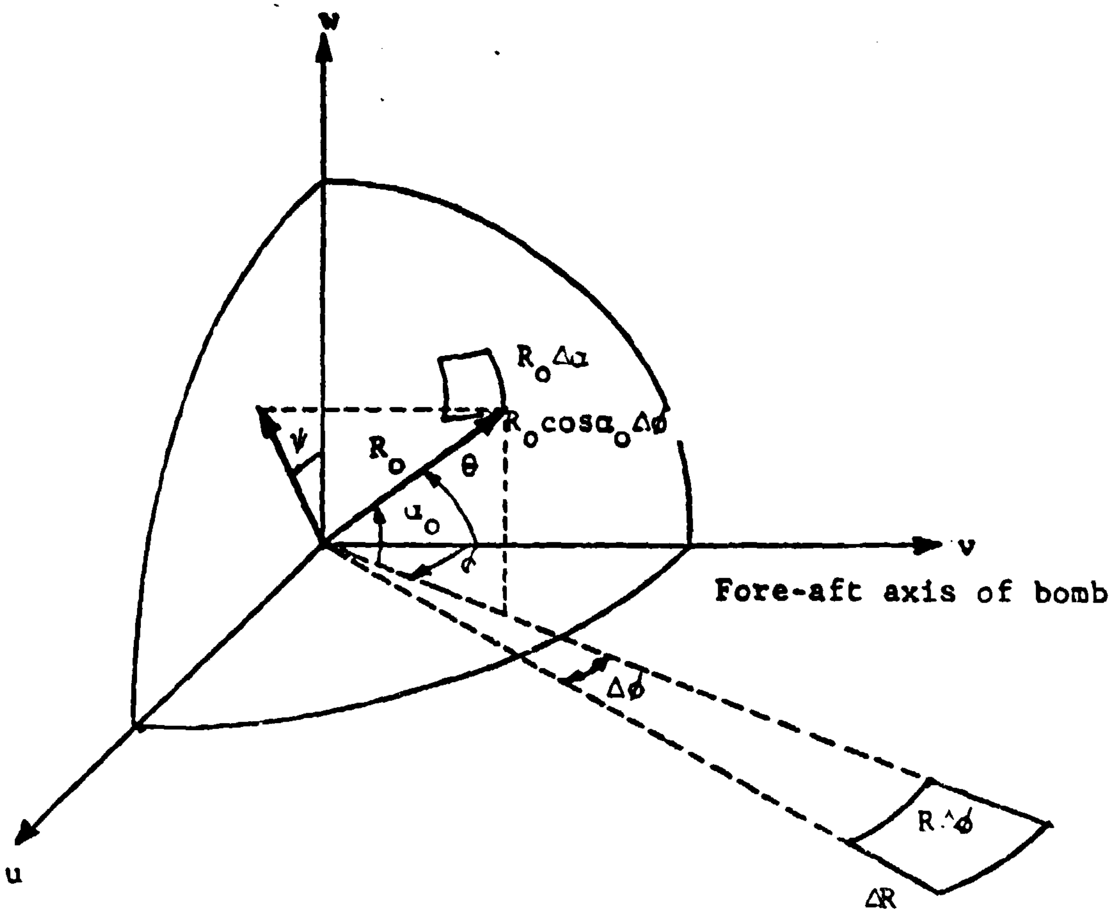


Fig. 3 FRAGMENT SOURCE GEOMETRY

Transformation formulas between the spherical coordinates (θ,ψ) with the munition nose at the pole and the spherical coordinates (α_o, ϕ) with the pole vertical give the following relations:

$$\cos\theta = \cos\alpha_o \cos\phi \tag{22}$$

$$\cos\psi = \sin\alpha_o/\sin\theta \tag{23}$$

Fragments on ballistic trajectories in the absence of wind travel in planes of constant azimuth ϕ.

For an individual munition, the initial fragment properties are functions of polar angle θ only. Therefore, for given elevation angle α_o and azimuth ϕ, the associated polar angle can be computed from Eq. (22) and all initial fragment properties at that point on the hemisphere can be obtained from munition performance data.

Now consider the fragments of a single mass interval, all having the same average ballistic properties, emitted from a particular munition at a point (α_o, ϕ) on the enclosing hemisphere. For simplicity, consider target points in the ground plane of Fig. 3. The terminal point corresponding to this initial point on the hemisphere is uniquely determined for each mass category through the associated ballistic trajectory.

The family of trajectories for fragments of a given average mass (i.e., those in one mass interval) can be thought of as a mapping of points on the hemisphere into points on the ground plane. Fragments of one mass interval originating from an element of area $R_o^2 \cos\alpha_o \Delta\phi\Delta\alpha_o$ on the hemisphere are projected into the element of area $R\Delta\phi\Delta R$ on the ground plane. Thus if n_i^o is the number density of fragments in mass interval i at the source hemisphere and n_i is the number density at the terminal point R, we have

$$n_i = n_i^o \frac{R_o^2 \cos\alpha_o}{R\,|dR/d\alpha_o|} \tag{24}$$

This formula permits computation of the number density of fragments in each mass interval originating from a point on the source hemisphere at any azimuth ϕ, since for given initial ballistic properties the terminal point R is known from trajectory calculations. To calculate the number density across a plane normal to the final trajectory, equation (24) is replaced by

$$n_i = n_i^o \frac{R_o^2 \cos \alpha_o}{R\left|-\sin \alpha_f \frac{dR}{d\alpha_o}\right|} \qquad (25)$$

where α_f is the final trajectory elevation angle.

For an individual munition the mapping derivative $dR/d\alpha_o$ is given by

$$\frac{dR}{d\alpha_o} = \frac{\partial R}{\partial \alpha_o} + \frac{\partial R}{\partial V_o} \frac{\partial \theta}{\partial \alpha_o} \frac{dV_o}{d\theta} \qquad (26)$$

where V_o is the initial velocity. From Eq. (22) we have

$$\frac{\partial \theta}{\partial \alpha_o} = \sin \alpha_o \cos \phi / \sin \theta \qquad (27)$$

The partial derivatives $\partial R/\partial \alpha_o$ and $\partial R/\partial V_o$ can be obtained analytically for lower register fragments; however, for upper register fragments, they must be obtained by numerical differentiation of a precalculated ballistic file. The analytic expressions for the lower register are

$$\frac{\partial R}{\partial \alpha_o} = \frac{2R - \bar{x}_o \cos\alpha_o}{\tan 2\alpha_o} - \frac{\bar{x}_o \cos \alpha_o}{\sin 2\alpha_o} - \frac{2R-\bar{x}_o\cos\alpha_o}{\tan \alpha_f} \qquad (28)$$

$$\frac{\partial R}{\partial V_o} = - 2(R-\bar{x}_o \cos\alpha_o + \bar{x}_o \sin\alpha_o/\tan\alpha_f)/V_o \qquad (29)$$

The derivative $dV_o/d\theta$ is determined from munition effectiveness data.

The computer model, moving along rays of azimuth in the horizontal plane, determines the terminal properties of lower register fragments at preselected radial increments of range

out to maximum range. At maximum range the corresponding elevation angle is noted and the remaining elevation angle up to 90 degrees is divided into equal increments. The model, utilizing the multi-step trajectory routine with a fixed drag coefficient of 1.28, calculates the terminal properties of the upper register fragments corresponding to each increment of elevation angle. The fragment number density can be determined exactly at prescribed terminal points for lower register fragments and, by interpolation, at these same points for the upper register fragments. These contributions are accumulated to give the total fragment density at the preselected radial distances. By symmetry it is only necessary to perform the analysis in half the horizontal plane to one side of the munition axis.

For a particular target, nondamaging fragments based on the applicable target vulnerability criterion are excluded from the accumulated number density. The number densities so calculated represent, in a statistical sense, expected values of the numbers of impacts per unit area. That is, these results, calculated deterministically as a function of position in the horizontal plane, are the numbers of damaging impacts per unit area to be expected on the average. To calculate the associated probability of damage (i.e., impact by one or more damaging fragments) to a particular target requires a statistical representation of the process, incorporating these expected values of fragment number density. In this study the target is a sphere of unit cross sectional area and fragments are considered to intersect this target across a plane normal to the final trajectory direction.

2.3 Target Damage Probability

Calculation of the probability of damage to a particular target requires a statistical representation of the process of impact by damaging fragments, involving the target area and the expected number of impacts per unit area. The number density of fragments is a function of position of the spherical target in the horizontal plane of interest.

Consider a target of projected area A_i normal to the trajectory of fragments of mass interval i at the associated terminal point. Let $\bar{n}_i$ be the expected number of impacts per unit area of fragments satisfying the applicable target vulnerability criterion. The impact process is assumed to be uniformly random in the immediate neighborhood of the point of interest. That is, impact by a damaging fragment is equally likely on all equal elements of area in the vicinity of the point. Statistically, we then have what is termed a Poisson process. The probability $p_i(j)$ of exactly j impacts on the target by damaging fragments of mass interval i is given by

$$p_i(j) = \frac{(\bar{n}_i A_i)^j \exp(-\bar{n}_i A_i)}{j!} \tag{30}$$

The probability of impact by none of the damaging fragments of mass interval i is therefore

$$p_i(0) = \exp(-\bar{n}_i A_i) \tag{31}$$

The target is assumed to be damaged if struck by one or more damaging fragments of any of the mass intervals. On this basis, the probability of damage to the target, q, is given by

$$q = 1 - \exp(-\Sigma\, \bar{n}_i A_i) \tag{32}$$

This last formula permits direct computation of the probability of damage to a given target at every point of the horizontal plane of interest from the accumulated number densities of damaging fragments and the associated projected areas. The target in this study was a standing man. The man was considered to present a projected area varying from 1.33 sq ft in plan to 9.0 sq ft frontal area. A_i was then calculated as the sum of the projections of both a vertical and a horizontal target on a plane normal to the final trajectory direction. The vertical target had an area of 9.0 sq ft and the horizontal target had an area of 1.33 sq ft.

2.4 Description of Model Output

Contour maps, describing each of the seven munitions of interest, listed in Section 1.2, were produced utilizing the computer model and are included as Appendix B. For each munition there are three contour maps: one describing the fragment flux (i.e., target hits per sq ft) due to all fragments; another describing the fragment flux of all fragments with terminal energy exceeding 58 ft-lbs; and a third which presents the probability of serious injury to standing personnel computed as described in the previous section.

Another set of figures is shown in Appendix C. Here again, there are three sets of figures per munition. These are simplified curves, derived from the corresponding contour maps, and represent fragment density and damage probability in certain sectors of the ground plane as functions of radial distance only. There are three of these fixed sectors: that is, 10-degree sectors in the base and nose of the munition and a third sector representing a peak side-spray. This third sector varied somewhat among all of the munitions, but was usually defined by a line of peak values appearing within $\pm$ 10 degrees of the 90-degree azimuth. This line of peak values did not usually include the outermost or lowest value contour. The average value of this contour in the side spray sector was determined by inspection from the maps.

Both the contours and the corresponding curves represent the fragment field from a minimum range of 250 ft out to the prescribed limit of effect (i.e., one hit per 6000 sq ft for fragment densities and a probability value of 0.0001). The minimum range is a function of a prescribed input variable and was chosen as 250 ft in this study. Since the results represent a single unit of munition and it is anticipated that future use of these results may be directed at multipie unit stacks, it seems reasonable to assume that the far fields results will be of more interest and the minimum range of 250 ft will be quite sufficient.

3. MUNITION EFFECTIVENESS DATA

Previous studies (Ref. 2) have indicated published munition effectiveness data, in their present form, are not suitable for far-field fragment hazard analysis. In general, these data place heavier fragments into one or two broad weight intervals with a corresponding average weight. Since the heavier fragments travel greater distances and their resulting terminal kinetic energy effect is high, it is necessary that these fragments be resolved into an adequate number of weight intervals.

This section of the report describes several tasks which were undertaken to provide a systematic approach to both illustrating the need for and subsequently the methods utilized in revising munition effectiveness tables.

3.1 Documentation of Original Fragmentation Arena Data

In collaboration with the Ballistics Research Laboratory at Aberdeen Proving Grounds, the fragmentation arena test procedure and data analysis techniques used to obtain munition effectiveness data were reviewed. Arrangements were made to examine the original arena test firing records at the APG Technical Library for the following munitions:

- 105 mm Howitzer Projectile M1
- 155 mm Howitzer Projectile M107
- 175 mm Gun Projectile M437A2
- 750 lb General Purpose Bomb M117A2 (APG Data)

Subsequently, contacts were established at NWL to obtain similar arena test data for the following munitions.

- 5-in/38 Projectile Mark 49 Mod 0 (VT)
- 8 in/55 Projectile Mark 25 Mod 1 (HC)
- 500 lb Low Drag Bomb Mark 82 Mod 1
- 750 lb General Purpose Bomb M117A2 (Eglin AFB Data)

For each munition the following information was documented.

- Test munition physical measurements
- Specifications for each arena test facility
- Listing of mass groups and polar zones
- Individual listing of fragment weights greater than 150 grains.

An understanding of the fragmentation arena test techniques is an important factor in the proper assessment of the larger and more hazardous fragments. The arena test is used to determine fragment mass, velocity and spatial distribution of high-explosive munitions. The munition axis of symmetry is located horizontally and is taken as the polar axis. Designating the polar angle by θ and the azimuth angle as ϕ, the fragmentation characteristics are a function of θ, and gravity effects are assumed negligible. An arena is constructed of an appropriate size as illustrated in Fig. 4. The arena test is designed to sample fragmentation characteristics in various polar angle intervals ranging between $\theta = 0$ degree at the nose, and $\theta = 180$ degrees at the base of the munition. These data samples are used to predict the fragmentation characteristics for the entire munition.

An important aspect of the arena test procedure is the relationship between the sample size obtained from the incremental polar zone on the arena recovery panel and the corresponding munition polar zone. Calculations were made from actual arena setup data to estimate the magnification factor associated with the integrated munitions data. These data, summarized in Table 2, indicate that only about 2 to 7 percent of the fragments are sampled in the 90 degree polar angle region (side spray) for each test. Thus, each fragment must be multiplied by a factor of 15 to 60 to obtain integrated data. It should be noted that the fragment sample size is about doubled at the 30 or 150 degree polar angle regions, and then increases rapidly to near 100 percent at 0 and 180 degrees (nose and base regions).

Table 3 summarizes data associated with fragments in excess of 150 grains for the munitions of interest. The 150 grain fragment was assumed to be the minimum damaging fragment mass projected at low elevations. Table 3 indicates that fragments in excess of 150 grains represent only 5 to 18 percent of the

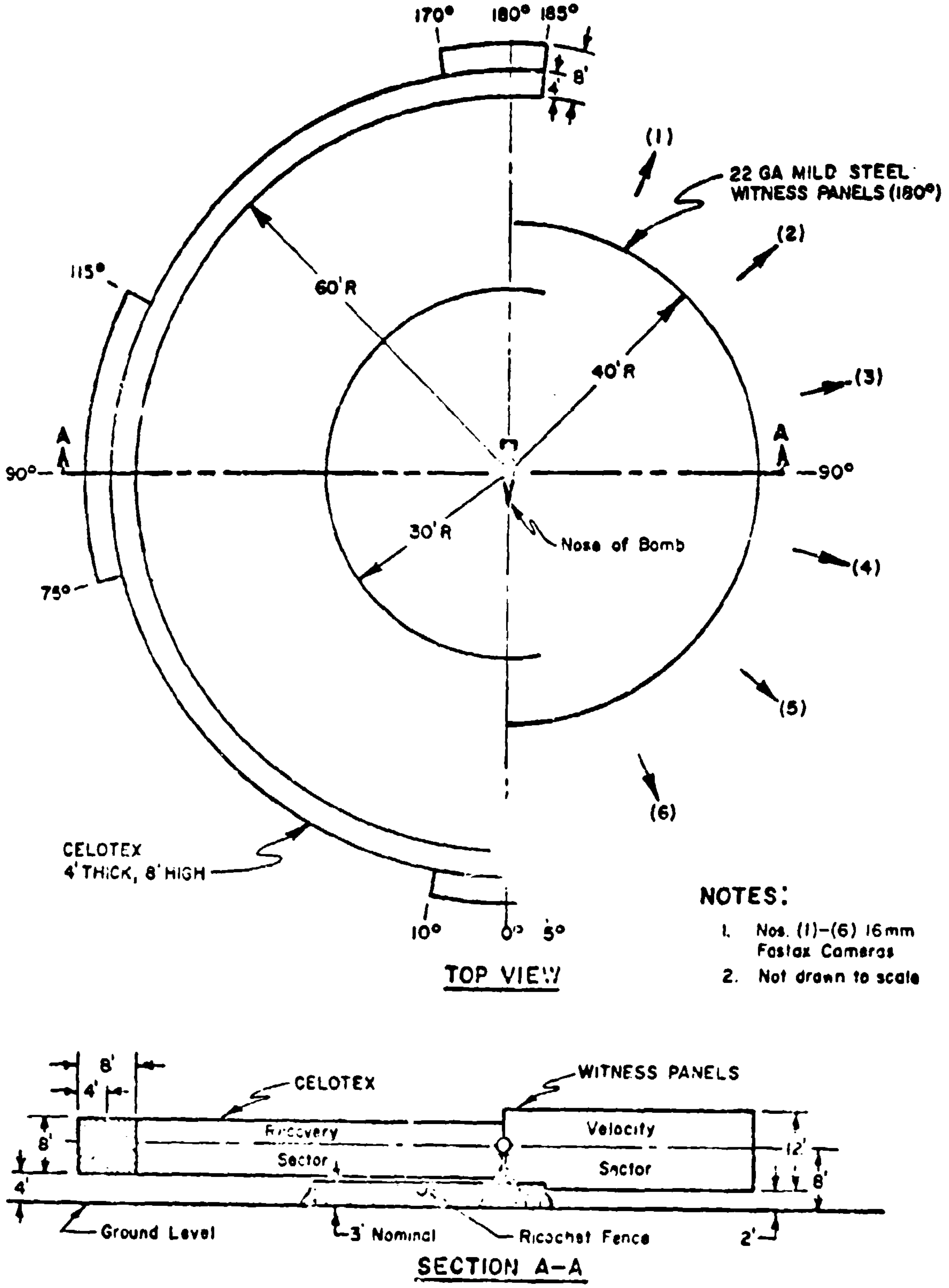
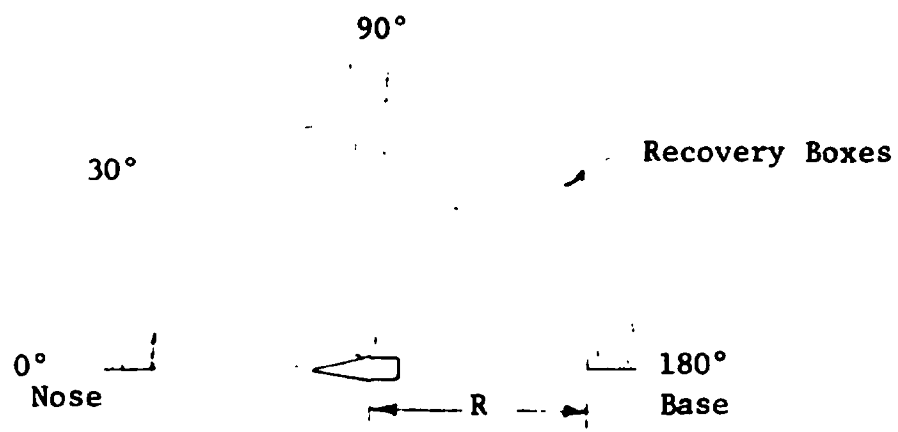


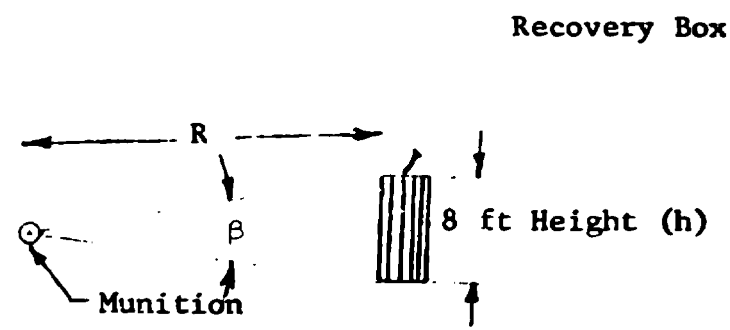
Fig. 4 FRAGMENTATION ARENA DIAGRAM (500 LB MK 82-1 BOMB)

TABLE 2 ESTIMATED FRAGMENT SAMPLE SIZE DATA FOR GIVEN ARENA TESTS

Munition Type	Arena Radius (R) (ft)	Recover Box Height (h) (ft)	90 Degree Polar Zone Region (Side Spray) Zone Angle β (deg)	Sample Size (β/360°) (%)	Mag. Factor (μ)
105 mm Projectile	20	8	22	6.1	16.4
155 mm Projectile	28	8	16	4.4	22.8
175 mm Projectile	31	8	14.5	4.0	25.0
5 in./38 Projectile	17	8	25	2.0	14.3
8 in./55 Projectile	40	8	11	3.1	32.3
500 lb Bomb	60	8	7.5	2.1	42.6
750 lb Bomb	75	8	6	1.7	58.8



Plan View of Arena



90° Polar Angle Arena Cross-Section

TABLE 3. DATA SUMMARY FOR FRAGMENTS GREATER THAN 150 GRAINS

Munition Type	Metal Weight (lbs)	Total No. of Fragments NT	Data for $M > 150$ Grains		
			N	%(NT)	% WT
105 mm Projectile	26.1	5,293	315	5.9	52
155 mm Projectile	77.8	6,994	943	13.5	70
175 mm Projectile	111.3	10,819	1364	12.6	78
5 in./38 Projectile	21.8	15,888	722	4.5	58
8 in./55 Projectile	231.0	8,661	1570	18.1	93
500 lb Bomb	320.9	22,114	2564	11.6	81
750 lb Bomb	339.0	24,148	3650	15.1	86

Notes: WT - Total Metal Weight Considered
NT - Total Number of Fragments (Calculated)
N - Number of Fragments Greater than 150 Grains (Approximate)
% NT - Percent No. of Fragments Greater than 150 Grains
% WT - Percentage of Total Weight for Fragments Greater than 150 Grains

total number of fragments but comprise 52 to 93 percent of the total case metal weight considered. In general, the weight groups beyond 150 grains are few in number and wide in weight interval. Since these heavier fragments are few in number and averaged over wide weight intervals, the data samples developed from arena tests are statistically inadequate in describing the characteristics for these potentially damaging fragments. As an example, the munition effectiveness data may list 60 fragments in one polar zone, each weighing 1100 grains and with a common velocity, while all other zones are free from that category of fragments. As input to the hazard model, these data could yield a ring-shaped hazardous region in the far field. It may be more reasonable to assume these fragments to be distributed statistically such as 30 units at 900 grains, 20 units at 1200 grains, and 10 units at 1500 grains, with the fragments being dispersed over three polar zones. In this regard, the distribution characteristics associated with fragments less than 150 grains could be used to predict the statistical distribution characteristics for the heavier fragments.

It should be emphasized that the munitions effectiveness data are valid statistically for characterizing munition performance for fragments of interest to the user. In general, the heavier fragments are only considered in the arena test procedure to assure that a conservation of weight is maintained between the recovered fragments in the arena sample and the integrated munition effectiveness data.

3.2 Revision of Munition Effectiveness Data

Seven sets of data, corresponding to the seven munitions of interest, were considered and of these, six were altered to overcome one of the following two fundamental deficiencies:

1) Gaps - Toward the heavier mass categories, "data voids" appeared making it difficult to approximate the cumulative distribution of mass as a continuous function.
2) Refinement - The cumulative distribution of mass loses definition where the final mass categories are an amalgam of all mass greater than the upper limit of the preceding category.

The first of these deficiencies were more predominate in the 175 mm and 155 mm munitions while the remaining sets of data, with the exception of the 105 mm, had poor refinement in the heaviest mass category. Each problem was handled differently, and in general, the problem of refinement was the most amenable since it was not necessary to add "fictitious" mass categories as it was in the case of data with void deficiencies.

It is prudent to comment that the best improvement which can be made on the experimental data is increasing the sample size. It is recognized then that with restricted data collection techniques and limited sample sizes, extrapolation and statistical inference must be conservative and cautious.

The basic indicators which were used to establish trends and to make decisions, were the average mass ($\bar{m}$) per mass category, the associated standard deviation (σ), the ratio $\sigma/\bar{m}$, and the average mass frequency (f) for each mass category.

The primary data for a particular munition was processed to print out a matrix of fragment weights and frequencies with mass categories as columns and polar zones as rows. Calculated information consisted of average mass $\bar{m}_i$ over all polar zones j within each mass category i, the corresponding average frequency f_i, the standard deviation σ_i, a ratio $\sigma_i/\bar{m}_i$ for each mass category, the total weight $W = \Sigma\Sigma\ \bar{m}_{ij} f_{ij}$, and the cumulative distribution of weight

$$F_k = \sum_{i=1}^{k} \frac{\Sigma\ \bar{m}_{ij} f_{ij}}{W} \quad k = 1,2 \ldots\ldots N \tag{33}$$

for all N mass categories. The two fundamental deficiencies noted in the data were identified by inspecting the cumulative frequency curve for obvious gaps in the domain of the function and by observing anomalies in trend of standard deviation values for the mass categories.

Two cases which will serve to illustrate the method.

A. The 155 mm Munition

The cumulative distribution curve (Fig. 5) was plotted and it was noted that although the average mass categories range up

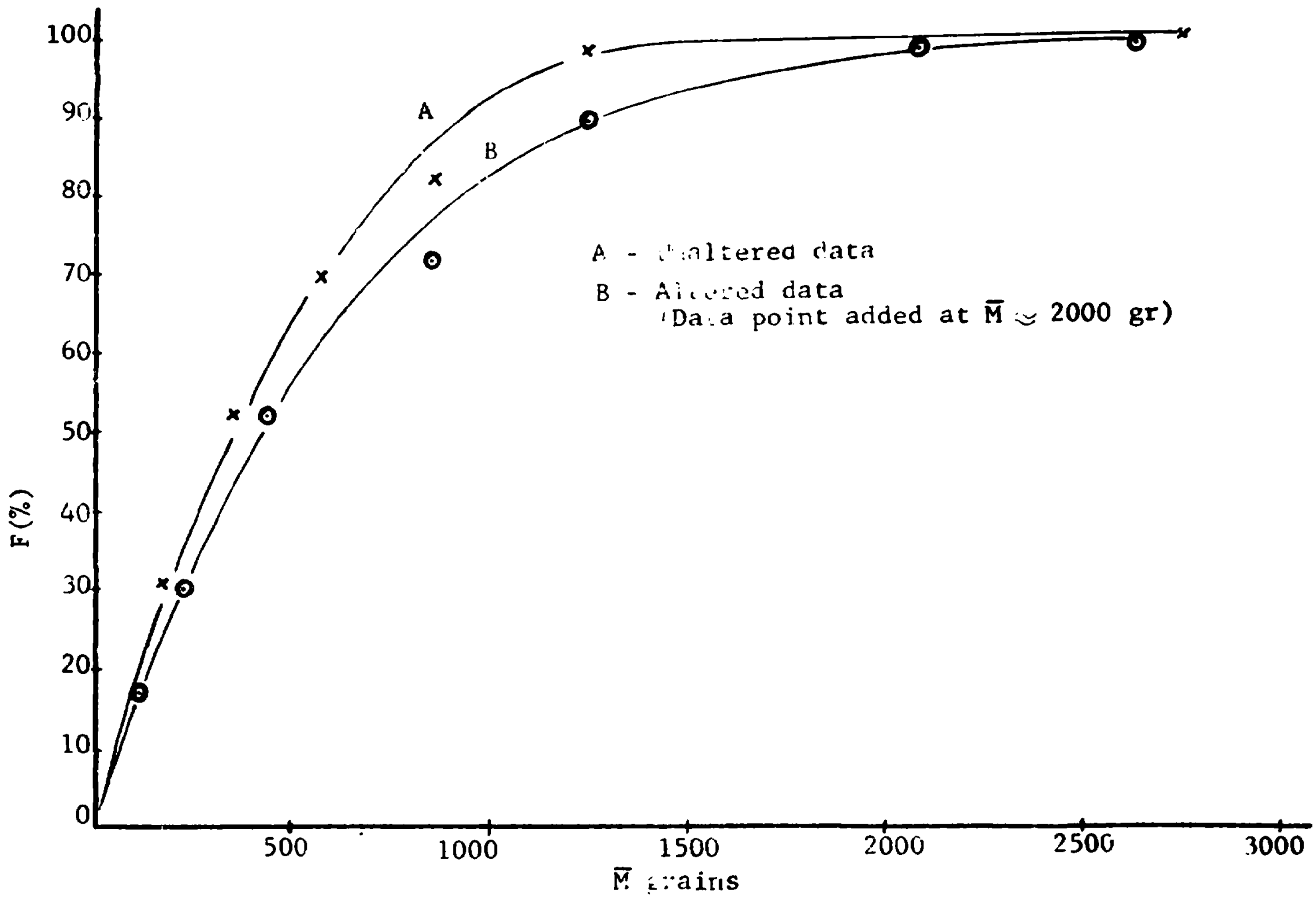


Fig. 5 CUMULATIVE DISTRIBUTION OF MASS FOR THE 155 MM MUNITION

to about 2800 grains, almost 100 percent of the weight has been accumulated at about 1200 grains. By inspection of the curve in Fig. 5 it was decided to start by adding a data point at about 2000 grains so that that portion of the curve would be less sparcely populated. The next decisions must phrase an answer to the questions: how much, and in what manner?

First and second moments were used as the indices of central measure and dispersion of the distribution of mass and frequency across the polar zones within a mass category. In general, the data available in the higher mass categories is not sufficient to construct a reasonable frequency histogram. In the lower mass categories it becomes possible to do this and in order to obtain a graphical sense for the distribution of this mass, histograms were constructed for several categories. Figure 6 depicts the weight distribution in the 200-250 category for the 155 munition. There is another distribution which will be used as a decision aid. This is the frequency of mass over all mass categories with polar zones constant. This curve is shown in Fig. 7.

Essentially then, we have a two-dimensional polar zone - mass frequency matrix in which we would like to make additional entries. Toward this end trends are identified in two directions: across columns (mass categories), and across rows (polar zones). The intersection of these trends at the point in question allows an estimate of the permissible entries which can be made at this point.

It was noted that the ratio $\sigma_i/\overline{m}_i$ tended to be fairly constant and, in case of the 155 mm munition, the average ratio was on the order of .061. Using this in conjunction with the intention of adding mass at $\overline{m} = 2000$, the dispersion measure is

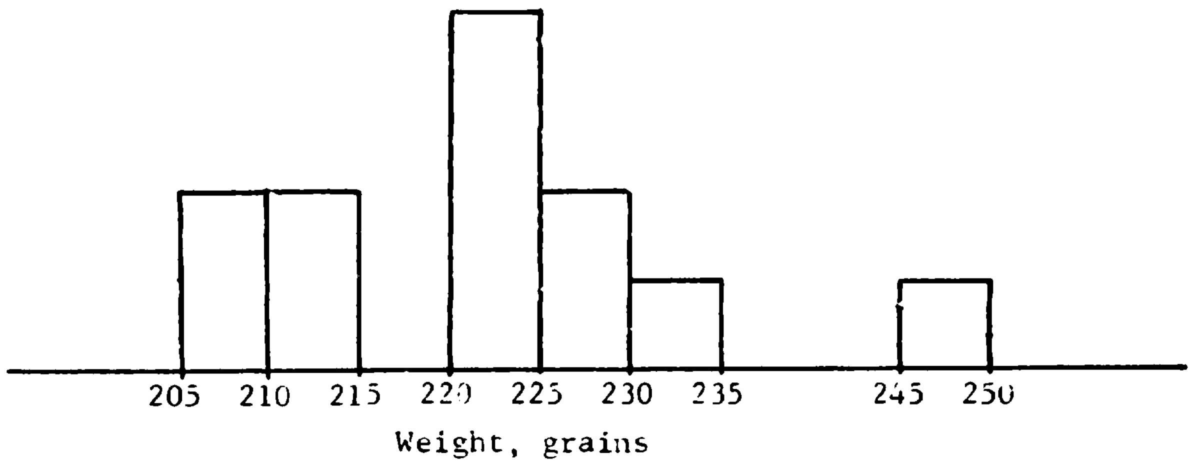


Fig. 6 DISTRIBUTION OF MASS IN THE CATEGORY 200 - 250 GRAINS FOR THE 155 MM MUNITION

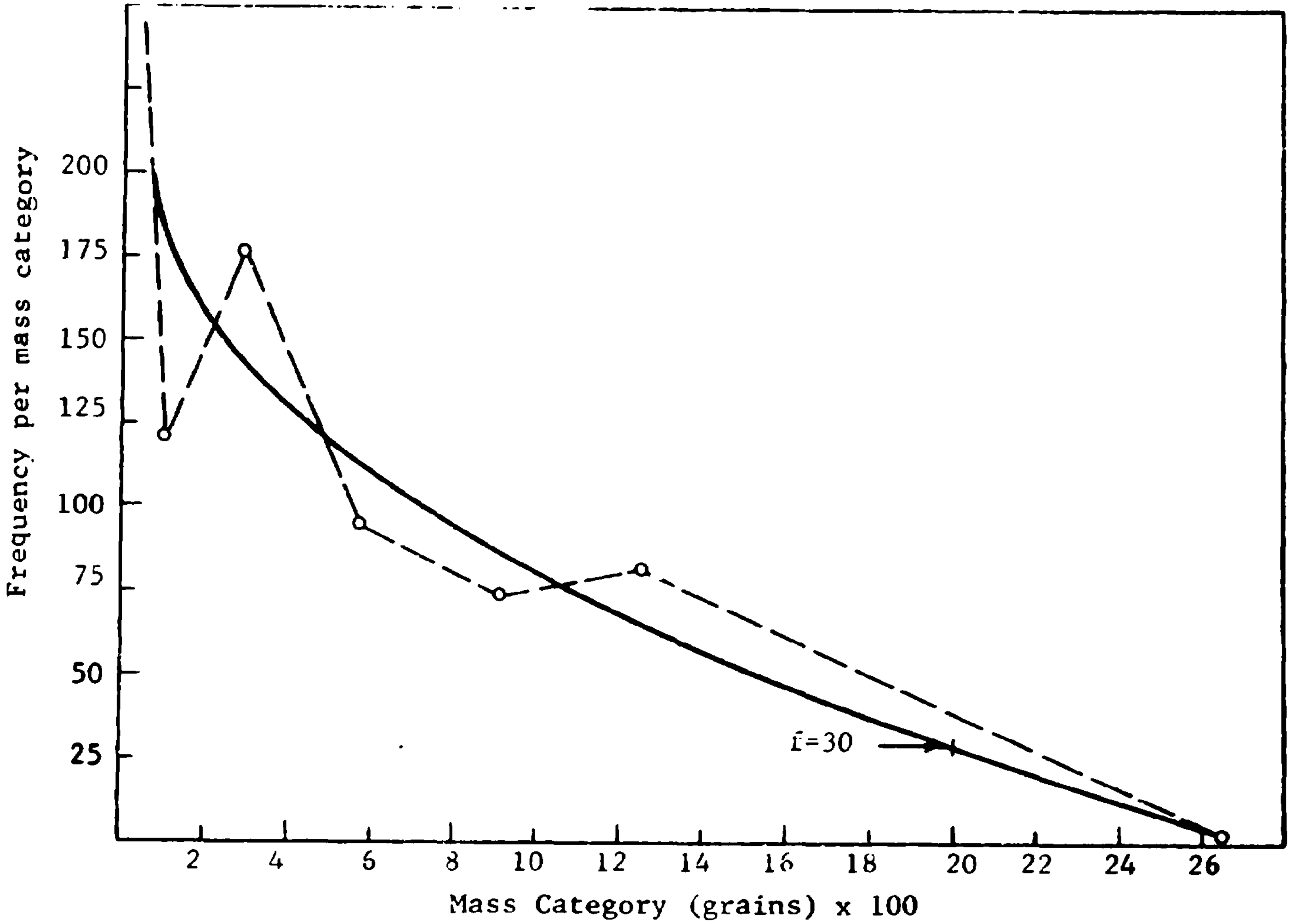


Fig. 7 POLAR ZONE FREQUENCY VERSUS MASS CATEGORY

estimated as .061 x 2000 = 122. The plot of polar zone frequency versus mass category (Fig. 7) was consulted, and a frequency of 30 was associated with the average mass of 2000 grains. By observing the trend across the mass categories, attention focused on polar zone 7 (60-70 degrees) and polar zone 11 (100-110 degrees). Further, by noting the distribution of frequencies across the polar zones in the previous categories, the same proportions are used to obtain:

Mass Category $\bar{m}$	=	2000 grains
Frequency f	=	30
Polar zones receiving entries	=	7 and 11
The frequency in previous mass category $\bar{m} = 1230$	=	79
Distribution of mass in zones 7 and 11 for $\bar{m} = 1230$	=	11/79 for zone 7 68/79 for zone 11
Distribution of mass in zones 7 and 11 for $\bar{m} \approx 2000$	=	$11/79 \times 30 \approx 4$, zone 7 $68/79 \times 30 \approx 26$, zone 11

Now using σ estimated at 122 for $\bar{m} = 2000$, we obtain

$$\bar{m} + \sigma = 2122$$
$$\bar{m} - \sigma = 1878$$

It should be noted at this point that the selection of $\bar{m}$, σ and consequently $\bar{m} \pm \sigma$ impose an implied condition of symmetry for the distribution of mass in that particular $\bar{m}$ category. This mass is further apportioned so as to agree with the trend generally defined by the preceding adjacent category thus bending the implied symmetry to conform with a distribution which is most likely not symmetrical. When data is added in this way the recalculated average mass will not generally agree with the original average mass $\bar{m}$, which was a somewhat arbitrary selection to begin with. The new $\bar{m}$ will be fairly close and in case of the 155 munition the data point was added at 2090 instead of $\bar{m} = 2000$. The completed profile for the new "fictitious" mass category consists of the following:

1. All polar zones empty except 7 and 11

2. A mass of 1878 grains with a frequency of 4 is entered in polar zone 7

3. A mass of 2122 grains with a frequency of 26 is entered in polar zone 11.

To complete the addition of this new mass category we now consider the conservation of total mass effects. Obviously after the addition of a new mass the total mass T_1 has been increased by some increment T_2. Thus, we have from the original or primary data

$$\Sigma\Sigma\, f_{ij}\, m_{ij} = T_1 \tag{34}$$

where m_{ij} is the fragment mass in the jth polar zone in the ith mass category, and f_{ij} is the associated frequency. Now with the added data points we have

$$\Sigma\Sigma\, f_{ij} m_{ij} + \Sigma\Sigma\, f^*_{ij}\, m^*_{ij} = T_1 + T_2 \tag{35}$$

where m^*_{ij} and f^*_{ij} are the added data for the additional mass categories $\overline{m}_i$. The objective is to scale down all f_{ij}, and f^*_{ij} proportionately such that for new f_{ij},

$$\Sigma\Sigma\, f_{ij} m_{ij} + \Sigma\Sigma\, f^*_{ij}\, m^*_{ij} = T. \tag{36}$$

Toward this end, let p_{ij} be that proportion of f_{ij} which generate the new f_{ij} such that Eq. (36) is satisfied for new f_{ij}.

Then for original f_{ij},

$$\sum_{i=1}^{k}\Sigma\, p_{ij} f_{ij} m_{ij} + \sum_{i=k}^{N}\Sigma\, p_{ij} f^*_{ij} m^*_{ij} = T_1 \tag{37}$$

Dividing both sides of (37) by $T_1 = \Sigma\Sigma\, f_{ij} m_{ij}$, we have

$$\frac{\Sigma\Sigma\, p_{ij} f_{ij} m_{ij}}{\Sigma\Sigma\, f_{ij} m_{ij}} + \frac{\Sigma\Sigma\, p_{ij} f^*_{ij} m^*_{ij}}{\Sigma\Sigma\, f_{ij} m_{ij}} = 1 \tag{38}$$

recalling that $\overline{x} = \dfrac{\iint x\, f(x)\, g(x)}{\iint f(x)\, g(x)}$

the first term in (38) is $\overline{p}$. Hence,

$$\overline{p} + \frac{\sum\limits_{i=k}^{N}\Sigma\, p_{ij} f^*_{ij} m^*_{ij}}{\sum\limits_{i=1}^{k}\Sigma\, f_{ij} m_{ij}} = 1 \tag{39}$$

Since the sum $\Sigma\Sigma\, p_{ij} f^*_{ij} m^*_{ij}$ is over small ranges of i and j the error will be small if we substitute $\overline{p}$ for p_{ij}

$$\overline{p} + \overline{p} \; \frac{\Sigma\Sigma\, f^*_{ij} m^*_{ij}}{\Sigma\Sigma\, f_{ij} m_{ij}} = 1 \tag{40}$$

or $\quad \overline{p} + \overline{p}\,(T_2/T_1) = 1$

and

$$\overline{p} = \frac{T_1}{T_1 + T_2} \tag{41}$$

The result shows that we can readjust all the frequencies by one constant of proportionality. Hence, a complete set of new data is generated with an added data point on the cumulative curve with the conservation of mass. The new cumulative curve, plotted from the revised data set, is shown in Fig. 5. As in every other case, the curve is shifted downward, which introduces a conservative element into the interpretation of the data.

B. The 500 lb Bomb

The cumulative distribution for this munition was plotted from the original data and is shown in Fig. 8. Without modification, the original data was divided into ten mass categories:

0-10, 10-20, 20-40, 40-80, 80-120, 120-150,
150-190, 190-230, 230-310, 310 + grains

The problem in this case is poor resolution. The standard deviation for example is 3.26 for category number 4, 8.33 for category number 6, 7.93 for category number 8 and 698.5 for category number 10. The standard deviation of 698.5 is not in itself unexpected. Because of its magnitude, it is an anomaly since its ratio $\sigma/\overline{m}$ = .754 is more than ten times the expected value of approximately .06. Moreover, it is a sharp break in the standard deviation trend. If we focus on mass category No. 10, then we find it labeled as 310 +. This is misleading since, in fact, mass category number 10 contains fragments which weigh as much as 5000 grains. To overcome this resolution deficiency, the 10th category was expanded into five additional categories such that the total number of mass categories became 15.

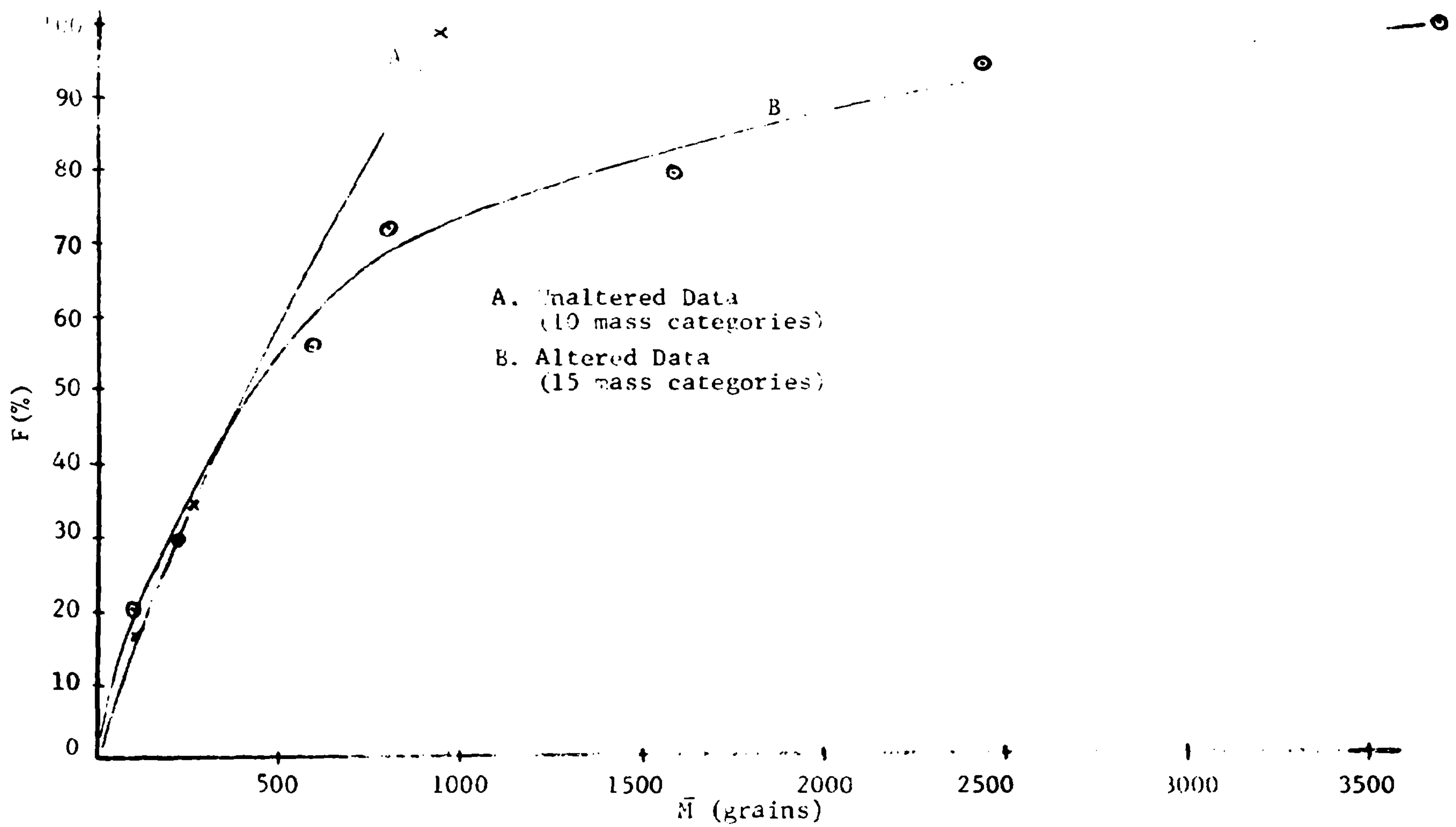


Fig. 8 CUMULATIVE DISTRIBUTION FOR 500 LB BOMB

Category	Range
10	310-450
11	450-750
12	750-1000
13	1000-2000
14	2000-3000
15	3000 +

These mass categories were then added on to the data, replacing the former category number 10. The result was a better than 50 percent improvement in the average $\sigma/\overline{m}$ for categories 10 thru 15 and a smoothing of the expected standard deviation trend. The altered cumulative distribution is shown in Fig. 8.

The data received from the Navy was not directly usable with out extensive recoding and key punching. This was done for the 750 lb bomb and the cumulative distribution appears in Fig. 9. This curve falls below the curves for the original data and its altered form. Since the lower curve implies more mass in the higher mass categories it also implies a greater damage influence range. In all cases the alteration of the data produced the same effect by lowering the cumulative distribution curve.

The following is a summary of how the remaining data were altered. No alterations were necessary for the 105 mm shell.

750 lb bomb: originally had 12 categories. The label on category 12 was 450+. Six additional categories were added as follows:

12 450-700
13 700-900
14 900-1200
15 1200-1500
16 1500-2000
17 2000-3000
18 3000 +

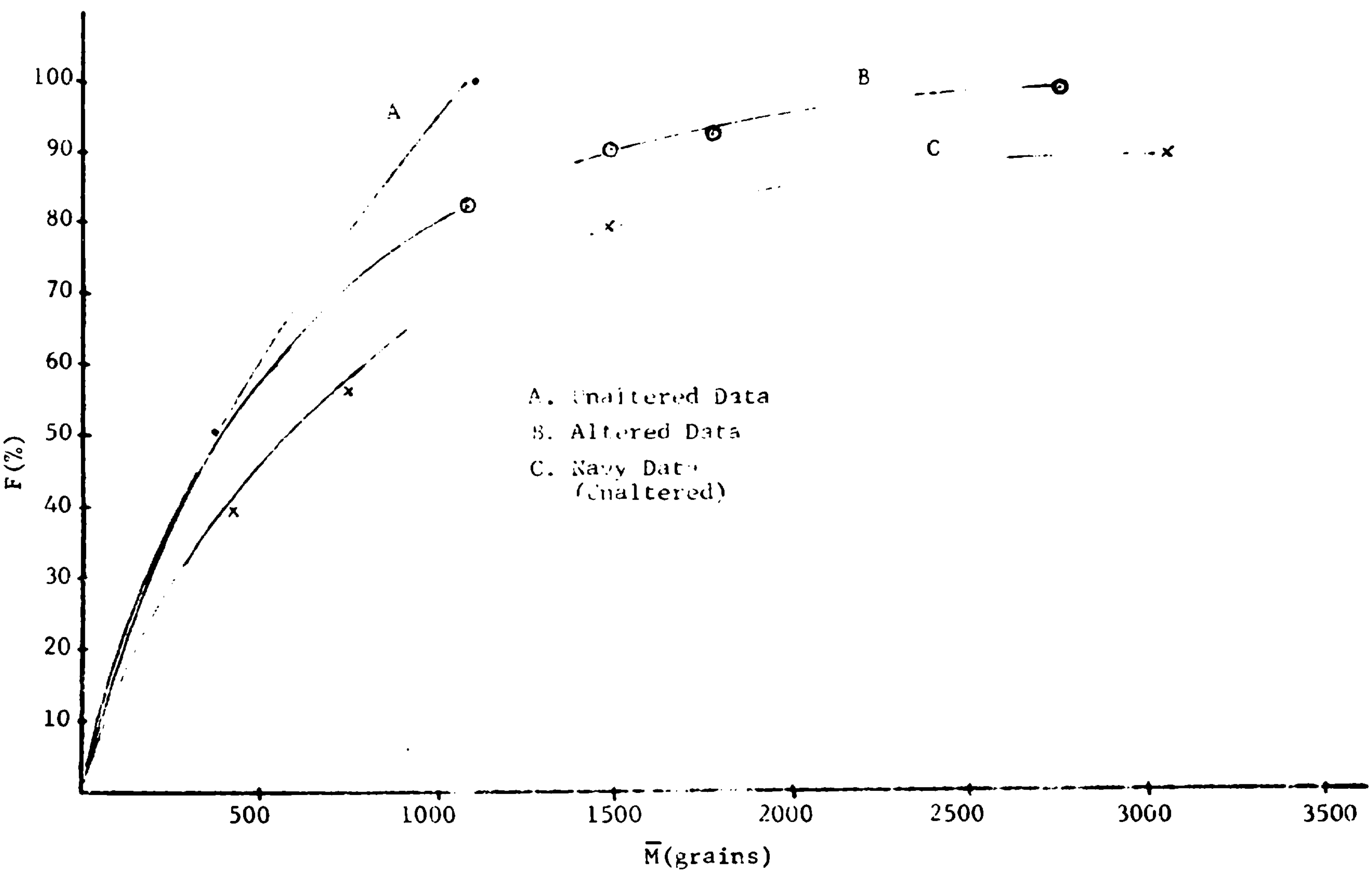


Fig. 9 CUMULATIVE DISTRIBUTION FOR 750 LB BOMB

The cumulative distribution is shown in Fig. 9

5 inch Munition: 11 mass

categories were expanded to 14 as follows:

11	250-450
12	450-700
13	700-1500
14	1500-2000

The cumulative distribution is shown in Fig. 10.

The 8 inch Munition: 10 mass

categories were expanded to 16 as follows:

10	160-300
11	300-500
12	500-700
13	700-1000
14	1000-2000
15	2000-3000
16	3000+

The cumulative distribution is shown in Fig. 11.

The 175 mm munition: two data points were added to the cumulative distribution as shown in Fig. 12. The technique was the same as described for the 155 mm munition.

A complete set of the revised munition effectiveness tables are presented in Appendix E to this report. Their format is similar to that shown in Table 1.

3.3 Model Sensitivity to Input Data

In order to gain an appreciation for the sensitivity of the fragment hazard model output to alteration of munition effectiveness data, which is input to the model, a computational experiment was conducted. The unaltered and altered data sets for the 175 mm gun shell were used as input to the model. The results of these two cases are shown respectively in Figs. 13 and 14. The primary difference between the two sets of contours is the location of the lowest value contour line.

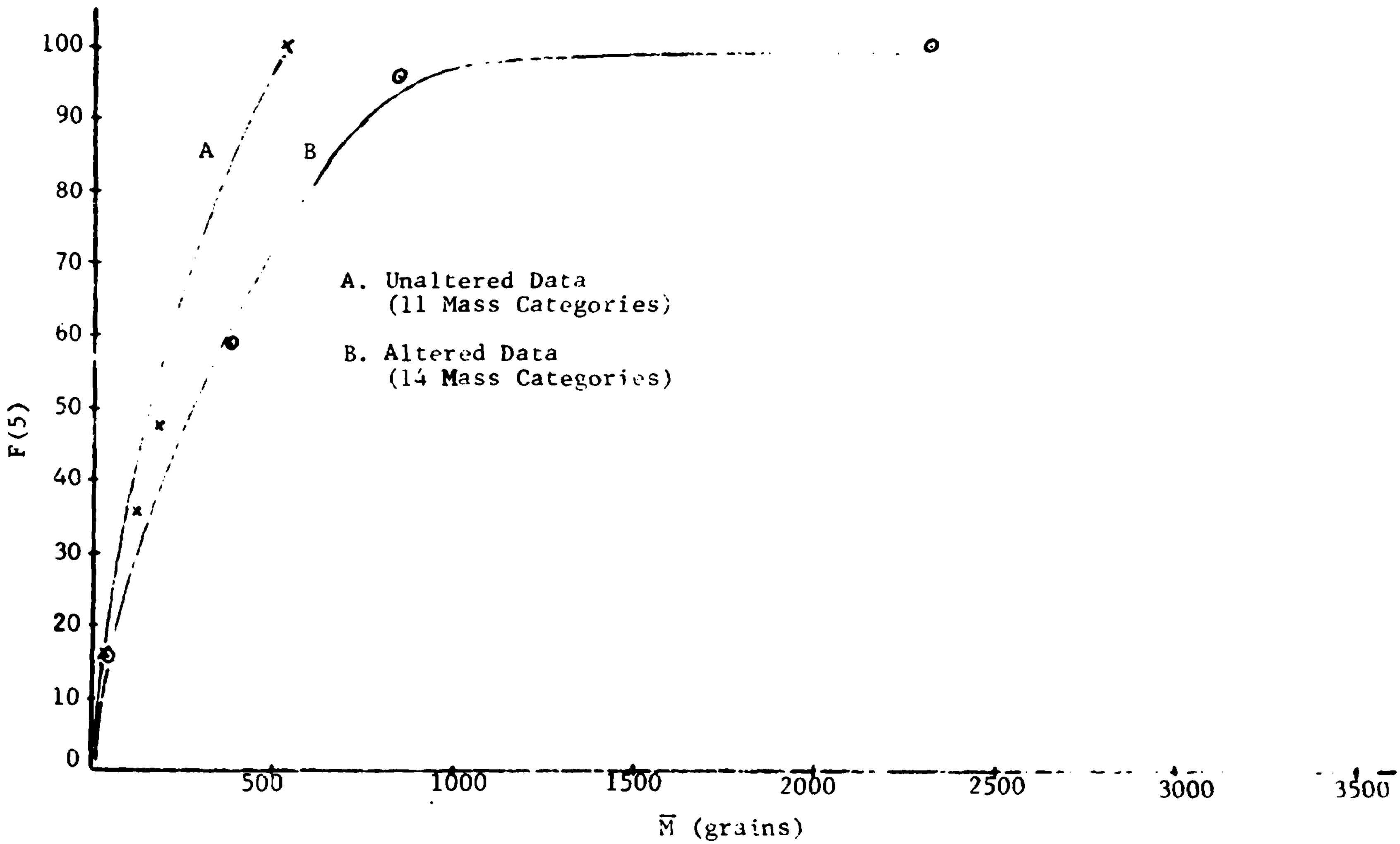


Fig. 10 CUMULATIVE DISTRIBUTION OF MASS FOR THE 5 IN. MUNITIONS

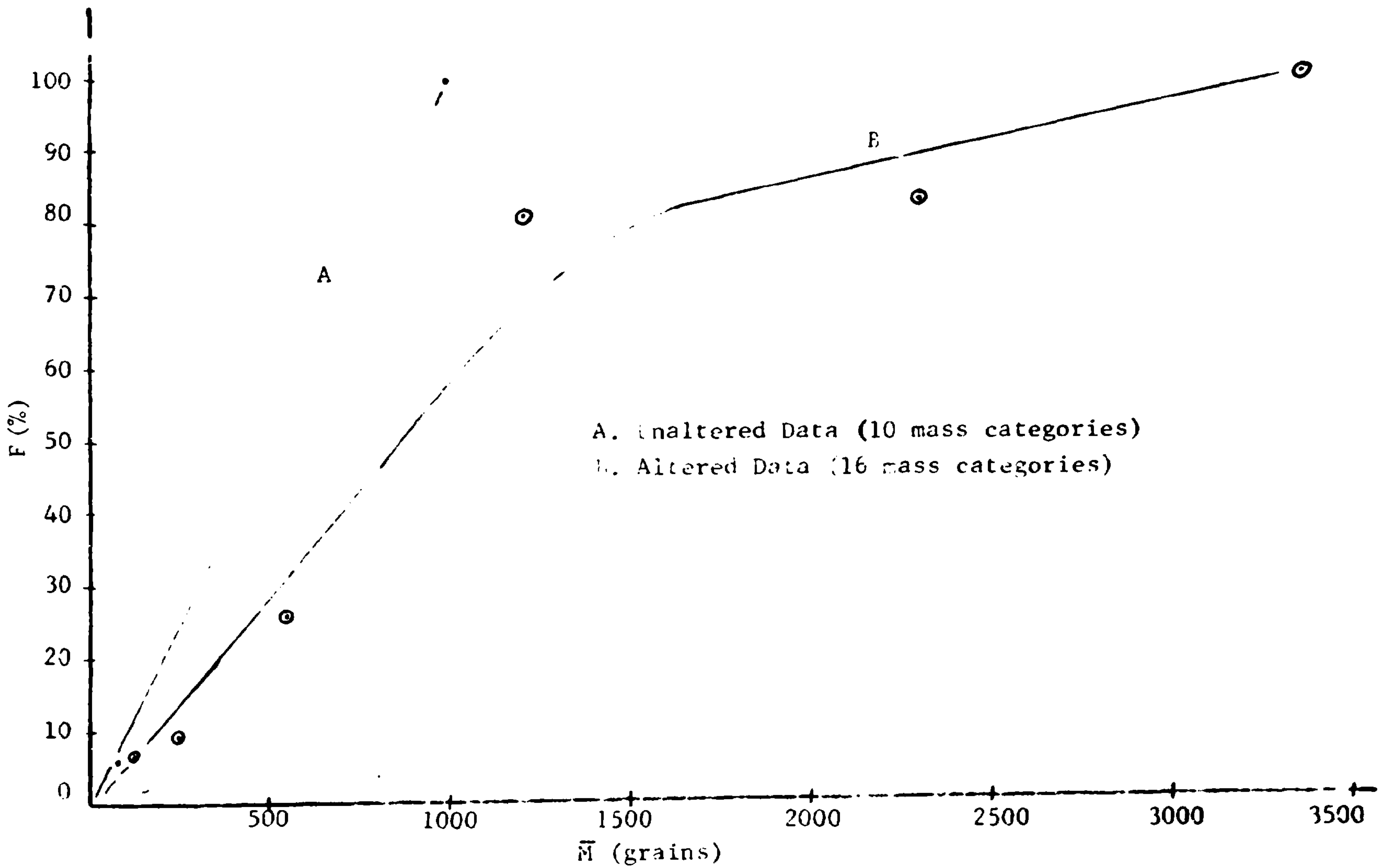


Fig. 11 CUMULATIVE DISTRIBUTIONS OF MASS FOR THE 8 IN. MUNITIONS

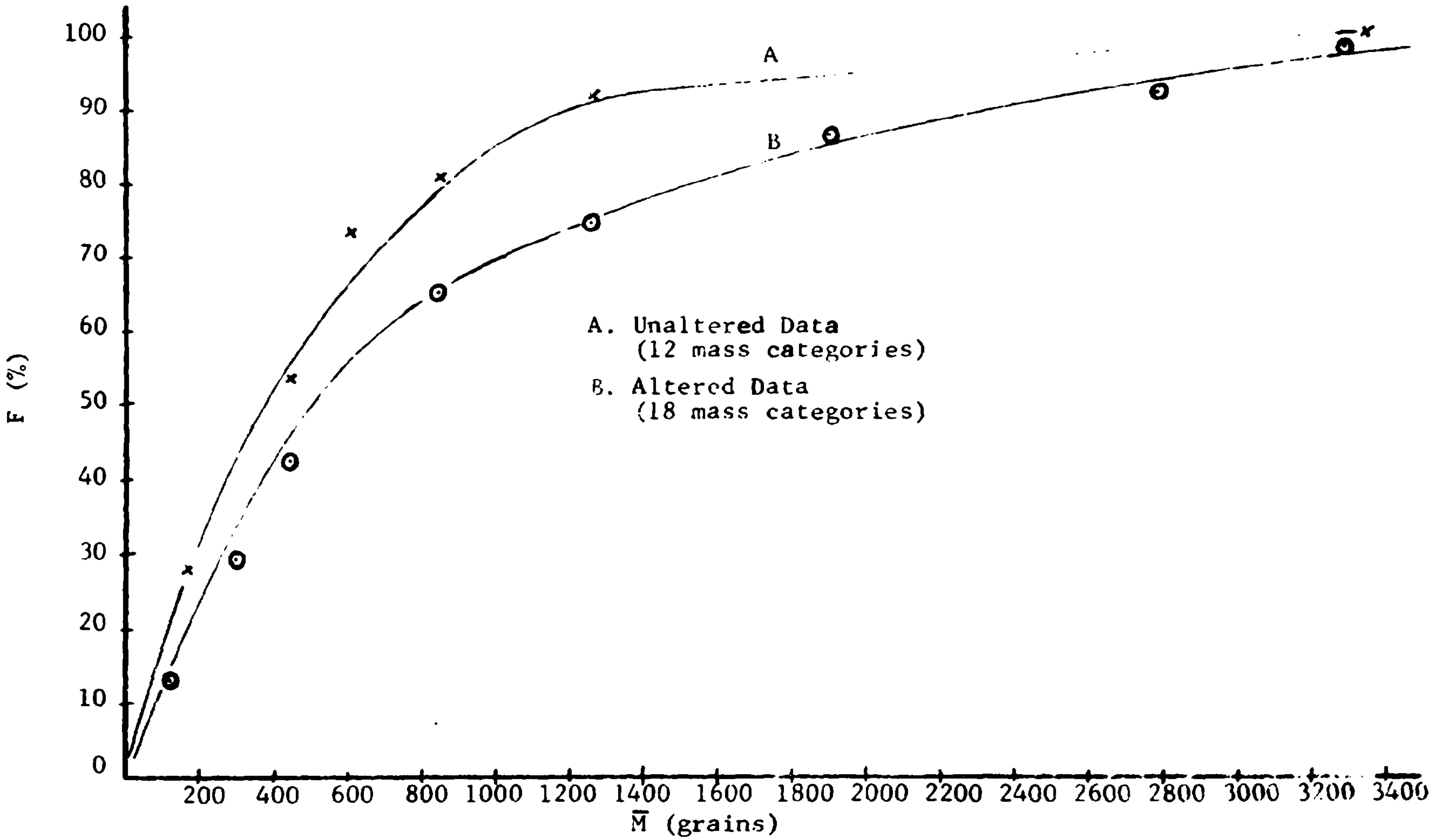


Fig. 12 CUMULATIVE DISTRIBUTION OF MASS FOR THE 175 MM MUNITIONS

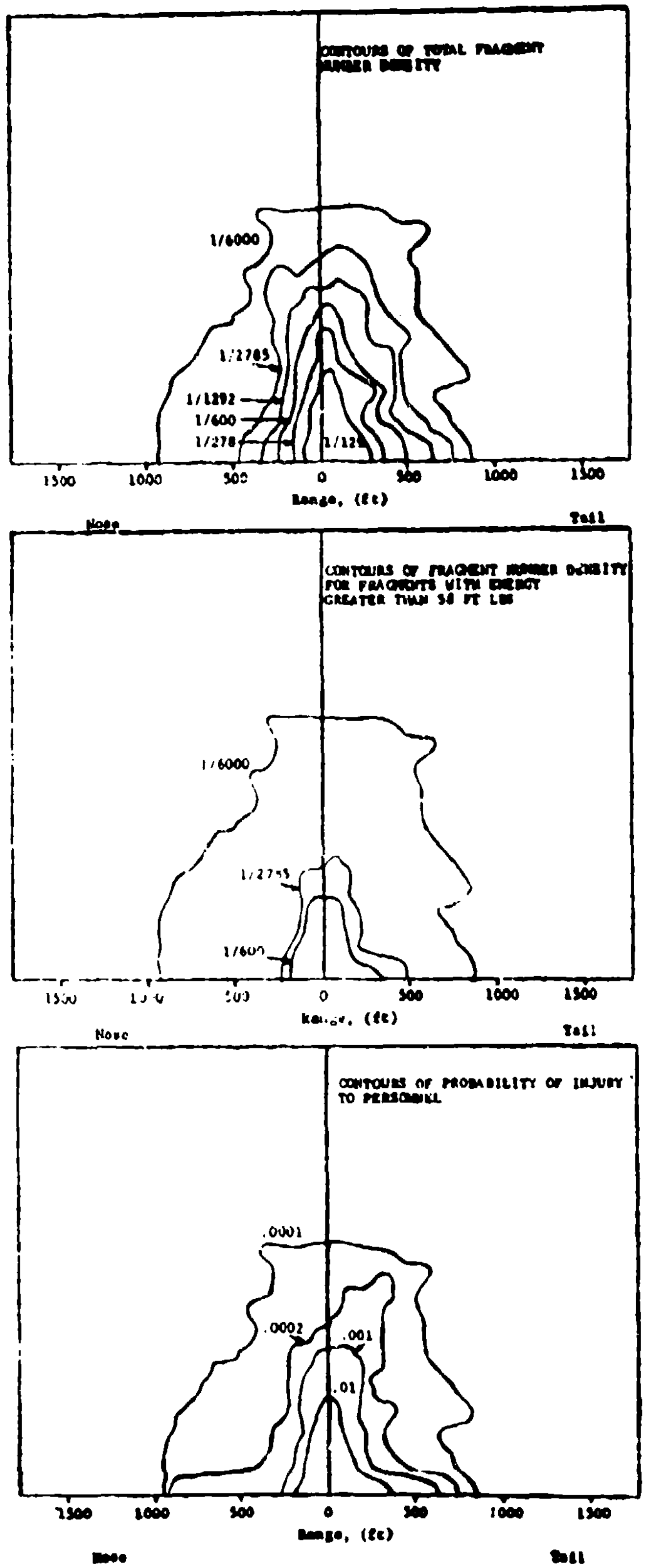


Fig. 13 175 MM GUN SHELL M437A2 (COMP. B LOAD) (UNALTERED DATA)

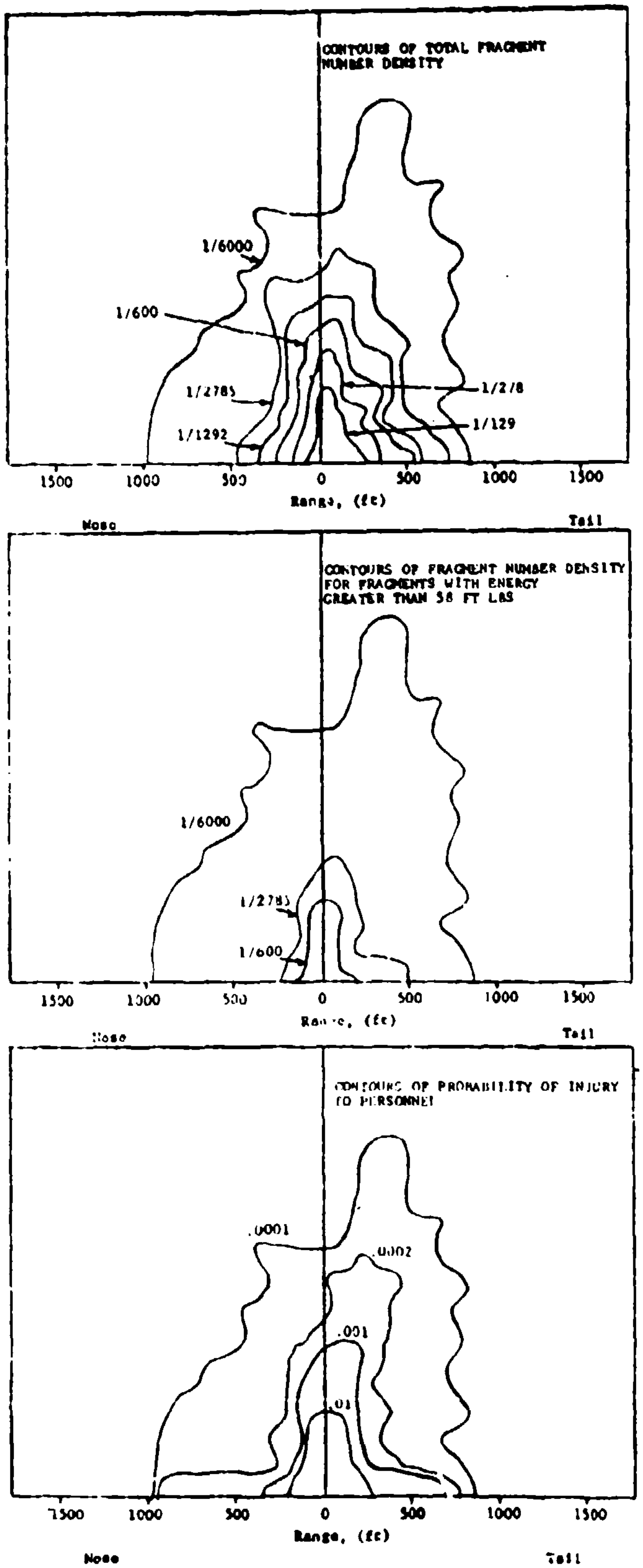


Fig. 14 175 MM GUN SHELL M437A2 (COMP. B LOAD) (ALTERED DATA)

The altered data set represents an emphasis on the resolution of the heavier fragments, (e.g., note Fig. 12) at the expense of lower weight fragments. This has the pronounced effect of shifting the location of the lowest value contour line outward for the altered data set. In terms of safety, then, the altered data set results in a more conservative set of contours.

4. RESULTS, CONCLUSIONS, AND RECOMMENDATIONS

This study has resulted in the development of a computer model which generates the information necessary in establishing minimum separation distances between various munition types and personnel in order to mitigate fragment hazards. The model specifically treats the fragment hazard associated with a single munition and has been utilized to generate single unit fragment hazard data for seven common military munitions. While single unit detonation does not represent a realistically severe accident situation, previous work indicates that multiple unit (i.e., stacks) fragment hazards may be proportional to single unit results. If this is true, and there is evidence to support such a hypothesis in the case of thin wall munitions, then the results obtained herein are directly applicable to estimating the fragment hazards associated with openly stored munitions.

4.1 Results

- Contour maps, expressing number density and injury probabilities (i.e., Appendix B), are seen to be particularly pronounced in three fixed sectors of the ground plane; that is, the nose, tail and side-spray sectors. Average values within each of these sectors have been expressed as functions of the radial distance from the explosive source. (i.e., Appendix C)

- Examination of the relationships in Appendix C indicate that the side spray sector is the critical area in the case of thick wall "shell type" munitions and to a limited degree in the thin wall "bomb" munitions.

- The curves in Appendix C also serve to illustrate how the kinetic energy criterion for personnel injury (i.e., 58 ft-lbs) reduces the applicable number density. At a given range the number density is reduced as much as an order of magnitude. It should be recognized that consideration of a less severe kinetic energy criterion will lead to lesser reduction in number density and thus a more conservative injury criterion in terms of safety.

- Table 4 summarizes the results available in Appendix C for each of the seven munitions considered. The table gives the computed distances corresponding to a fragment density of one hit in 600 ft^2 for all fragments and fragments with a terminal energy in excess of 58 ft lbs.

TABLE 4

SUMMARY TABLE FOR THREE PRINCIPAL DIRECTIONS AT 1 HIT PER 600 SQ FT (ENTRIES ARE IN RADIAL FEET)

Munition	All Fragments			Hazardous Fragments **		
	Nose	Side	Tail	Nose	Side	Tail
750 (1)*	440	1060	740	220	690	500
500 (1)	220	825	595	210	670	450
175 (1)	250	840	575	250	450	200
155 (1)	290	810	510	120	400	230
105 (0)	240	650	360	100	270	150
8 (1)	325	660	240	140	520	120
5 (1)	310	720	340	140	275	150

*(0) = Unaltered Data
(1) = Altered Data

**Hazardous fragments are defined as having in excess of 58 ft lbs kinetic energy

4.2 Conclusions

- By utilizing trajectory analysis in conjunction with stochastic treatment of experimental input data and damage functions, a mathematical model has been developed for estimating injury/damage contours for various single unit munitions.

- The mathematical model currently utilizes published munition effectiveness data which has been altered to give more emphasis to heavier fragments. It has been observed that this results in extending contour levels outward giving a more conservative result insofar as fragment safety is concerned.

4.3 Recommendations

Before making specific recommendations concerning the results generated in this study and future research efforts it is prudent to discuss the adequacy of munition effectiveness data as input to the model.

The results generated by the computer model are dependent upon and quite sensitive to the munition effectiveness data, which is input. This data was originally generated to support munition effectiveness studies and is the result of explosive tests of single unit munitions. It is the only known source of information concerning near-field estimates of munition fragment size, number and initial velocity. However, since it has been collected to be utilized in weapon effectiveness studies, it is primarily concerned with the fragments which are effective within the applicable range of the munition. This has normally led to a set of data which has a high degree of resolution, in terms of weight intervals, where the greatest number of fragments are concentrated. This unfortunately is at a rather low fragment weight (e.g., below 300 grains). The remaining fragment weight of the munition is quite substantial, but because it does not break down into very many fragments and is not always projected into the designed zones of munition effectiveness, its recorded resolution is usually quite poor.

Another inadequacy of recorded munition effectiveness data is concerned with its use in representing the basis for multiple unit munitions fragment hazard analysis. Here, the primary

concern is whether the munition fragment size, number and initial velocities will be similar for munitions in single and multiple units.

In the present study an attempt has been made to overcome the first of these deficiencies in a conservative manner. That is, existing munition effectiveness data have been altered to adequately resolve higher weight fragments in a statistical sense. Recently reduced data, developed at the Naval Weapons Laboratory, tend to support these alterations; however, the best way to resolve this problem would be the design and use of an experimental procedure aimed at obtaining arena data for hazard analysis.

The problem in assuming similar fragment characteristics for multiple and single unit stacks is a much more difficult problem to resolve. A number of detonation tests of stacked munitions have been conducted in the past where resulting fragments have been collected. In some cases the fragments have also been sized and number and weight distributions computed. Results of these studies indicate that thinwall "bomb type" munitions tend to fragment into similar size fragments for both multiple and single units. However, it is becoming quite apparent that this is not the case for thickwall "shell type" munitions. Here, fragment weight and number distributions from multiple units are quite different from single unit munitions.

In light of the input deficiencies enumerated above, the following recommendations are made:

- To take care in utilizing results obtained in this study for single unit munitions, in projecting fragment hazards associated with multiple units of similar munitions. This is especially true of "shell type" munitions.

- To compare the analytic results for the single unit 750 lb bomb obtained in the present study with experimental multiple unit results obtained in the NWC-China Lake Tests of March, 1970. Such a comparison will serve to validate our analytic procedures and the extension of our results for estimating multiple unit "bomb" fragment hazards.

- To characterize the resulting fragment number and weight distributions obtained from the June, 1970 155 mm shell tests at Yuma and the corresponding 155 mm shell tests at China Lake in December of 1971. These tests are expected to be dissimilar amongst both themselves and as compared to single unit results. However, such comparison will document these differences and the resulting distributions can be utilized by the analytic model to estimate the fragment hazard associated with both these cases.

REFERENCES

1. Zaker, T. A., et al., "Fragmentation Hazard Study, Phases I and II, IITRI Final Report J6176 for the Armed Services Explosives Safety Board under Contract DAHC-04-69-C-0056, April, 1970.

2. Feinstein, D. I., et al., "Fragmentation Hazard Study Fragment Hazards From Detonation of Multiple Munitions in Open Stores," Phase II, IITRI Final Report J6176 for the Armed Services Explosive Safety Board under Contract DAHC-04-69-C-0056, August, 1971.

3. Shaw, J. E., "A Measurement of the Drag Coefficient of High Velocity Fragments," BRL Report 744, (October 1950) (U).

4. Zaker, T. A., "Trajectory Calculations in Fragment Hazard Analysis," Minutes, Thirteenth ASESB Seminar, 101-115, September, 1971.

5. Weapons Characteristics Report, "Joint Munitions Effectiveness Manual/Air-to-Surface (U)," March 1967, (S).

APPENDIX A

PROGRAM USER'S MANUAL

APPENDIX A
PROGRAM USER'S MANUAL

The computer program for the fragment hazard model has been written in such a manner as to accept a problem-oriented input language. The following discussion describes, in detail, the various control and data statements required to execute the program. An example of the use of these statements in a sample problem is included.

Control Cards

All control cards start with column 1 = '$'. The remainder of the card is free format. Only columns 1-72 may be used for control text; columns 73-80 are not examined and may be used for anything.

Control Statement Form

Control statements are of the form:

name field1 field2 ... fieldn

where 'name' identifies the control statement, and the 'field's are parameters whose form depend on the particular control statement. 'name' and the 'field's are separated from one another by

1) one or more blanks, or
2) a comma, padded on either side by one or more blanks.

Examples:
```
$PRINT CON,DATA
$TAPE 4,SRCH=01100,BKSP
$STOP
```

If desired, comments preceded by a '$' may appear after the last field.

Example:
```
$STOP $ END OF RUN
```

If a control statement will not all fit on one card, it may be continued to additional cards in the following way:

1) place a comma after the last field on the first card.

2) start the second card with a '$' and a ',' (padding with blanks if desired).

3) place remaining fields after the '$' and ',' on the second card.

4) as many continuation cards as necessary may be used.

Example: $ORDNANCE FORMAT=2,REDUCE,REZONE,NORM,SMOOTH,
$,READ=3,LIST $ READ FROM TAPE 3

Parameter Form

Control statement parameters may be of three forms:

1) Positional: A value which must appear in its proper order on a control statement, as the first parameter, third parameter, etc.

Examples: 3, 14.197, T, 'TITLE A'

2) Flag: A name, or the two characters 'NO' prefixed to a name. A flag may appear anywhere in a control statement after the positional parameters (if any). A flag is associated with a logical value: this value is TRUE if the flag's name is specified, or FALSE if 'NO' plus the flag's name is specified.

Examples: LIST, NOLIST, BCD, BIN

3) Keyword: A name, followed by either a BCD (3-8 punch) or an EBCDIC (6-8 punch) equal sign (=), which may be padded with blanks, followed by a value.

A keyword parameter may appear anywhere in a control statement after the positional parameters (if any).

The variable represented by the name is set to the provided value.

Examples: ORDER=2, Z=5.5, FLAGS=XY, TITLE='TITLE A'

Value Form

There are four types of values:

1) Logical: a string of one or more characters, not containing any of the following: dollar sign ($), comma (,), equal sign (=), blank, or apostrophe ('). If a T appears in the string before an F appears, the value is TRUE. If an F appears before a T appears, or if neither an F nor a T appear, then the value is false.

Examples: T, .TRUE., TRUFFLE and TRUE are all TRUE.
F, .FALSE., AFTER, FALSE and NO are all FALSE.

2) Integer: one or more digits, optionally preceded by a + or - sign.

   Examples: 0, -1234, +77111

3) Real: a real number. Decimal point and exponent may be used. The exponent, if present, must be of one of the following forms:

   En, E+n, E-n, +n, -n

   where n is one or two digits.

   Examples: 0, -1234, +77111, 3.1416, 6.0238E+23, 1.-5, -.1+10, 51E6

4) Alphabetic: two forms are allowed.

   - a string of one or more characters, not containing any of the following: dollar sign ($), comma (,), equal sign (=), blank or apostrophe (').

     Examples: ABC, 123, MACH-1

   - a string of zero or more arbitrary characters, enclosed in apostrophes ('). Within the string, one apostrophe is represented by a pair of apostrophes. Either the BCD (4-8 punch) or EBCDIC (5-8 punch) apostrophe may be used.

     Examples: 'ARTHUR', 'WHICH WAY?', 'DON''T GO', 'TITLE A', '' (a null string)

Notation

When describing control cards, some possibly unfamiliar notation is used. This can be explained with an example:

$$\text{\$TAPE n}\left\{\begin{matrix}\text{,BCD}\\ \text{,}\underline{\text{BIN}}\end{matrix}\right\}\text{[,REW]}$$

The use of { } means that only one of the arguments listed may be used. If one of the arguments is underlined, that one will be assumed if none of the group are specified. (In the example, BIN is assumed unless BCD is used. BIN and BCD may not both be used.) The use of [ ] means that the enclosed argument is not required; it may be used or not used.

$PRINT[,CON][,DATA]

Print controls

- CON — Cause control cards to be printed.
- DATA — Cause data cards to be printed

$STOP

End of run; execution is terminated.

$TIME

Print elapsed time since beginning of execution, and since last $TIME statement.

$$\text{\$TAPE n}\left\{\begin{matrix}\text{,BCD}\\ \text{,}\underline{\text{BIN}}\end{matrix}\right\}\text{[,REW][,SKIP=k]}\left\{\begin{matrix}\text{, SRCH=}\alpha\\ \text{,XSRCH=}\alpha\end{matrix}\right\}\text{[,BKSP][,WEOF]}$$

Tape file manipulations. If more than one of the operations (REW through WEOF) are specified, the operations take place in the order REW,SKIP, SRCH or SXRCH, BKSP,WEOF, regardless of the order in which the operations are specified.

- n — Tape unit number
- BCD — Tape is formatted (BCD)
- BIN — Tape is unformatted (binary).

  If neither BCD nor BIN is specified, BIN is assumed.
- REW — Rewind unit n.
- SKIP=k — Skip k records on unit n.
- SRCH=α — Search unit n, starting at its present position, for a sentinel record whose key is α (1 to 8 characters). Terminate if no such sentinel is found before the end-of-file.
- XSRCH=α — Proceed as with SRCH, but if an end-of-file is encountered, rewind unit n and search the entire file. Terminate if end-of-file is again reached without finding the proper sentinel.
- BKSP — Backspace unit n one record.
- WEOF — Write a terminal sentinal record and an end-of-file mark on unit n.

$ORDNANCE FORMAT=n[,NSETS=n][,REDUCE][,REZONE][,NORM][,SMOOTH]
[,ORDER=n][,READ=n][,LIST][,KWIKPLOT][,SUMMARY]

Cause a title card and a set of ordnance data to be read.

- FORMAT=n — Ordnance format code. =1 or 2.
- NSETS=n — Number of ordnance components (fuze, casing, etc.) If ≤ 0 or omitted, NSETS=1 is assumed.

REDUCE	Halve the number of mass categories by combining each pair of categories.
REZONE	Transform data from N+1 sets of values at the boundaries of N polar zones to N sets of values at the N polar zone midpoints.
NORM	Normalize data, given as total number of fragments per mass category per polar zone, to number per unit area on a unit sphere surrounding the ordnance.
SMOOTH	Fill in gaps in data, and (if ORDER>0) apply a smoothing function to the ordnance data.
ORDER=n	Use an nth order Fourier series to smooth data (if SMOOTH is specified).
READ=n	Read ordnance data from unit n, a BCD card image unit. If n=5 or if READ=n is omitted, ordnance data is assumed to follow the $ORDNANCE statement.
LIST	List ordnance data in tabular form.
SUMMARY	List data summary information: total mass per mass category, per polar zone, etc.
KWIKPLOT	Produce printer plots of velocity curve, and of mass and number curves for each mass category, all as a function of polar angle.

An ordnance title card is read immediately after a $ORDNANCE statement is encountered. Columns 1-72 contain a description of the ordnance.

Two basic formats are defined for ordnance data (FORMAT=1 and 2). Within the context of each format, several data conditioning options are available (REDUCE, REZONE, NORM, SMOOTH, ORDER=n).

Ordnance data is read from cards if the READ field is omitted, or if READ=5. If a READ field specifies some other I/O unit, input is read from that unit in card image form. This unit is assumed to be properly positioned.

The first card (or card image) of a set of ordnance data is as follows, regardless of the type of ordnance data.

COLS	FORMAT	NAME	DESCRIPTION
1-5	I5	NMORD	Number of mass categories. If NMORD> 30 (maximum number of mass categories), REDUCE must be specified to cut the number of mass categories in half.
6-10	I5	NVORD	Number of measurement positions along a meridian from nose to tail. Measurements may be made either at the center or at the edges of equi-spaced polar zones. In the latter case, REZONE must be specified NVORD≤37.
11-15	I5	IDORD	Ordnance I.D. number 0≤IDORD≤99.
16			(ignored)
17	A1	IVO	Ordnance modification code. If left blank, IVO='0' is assumed.
18-20			(ignored)
21-30	E10.0	XKORD	Average fragment drag factor (grains/in.^{3}

Format 1 Ordnance Data

Data of format 1 consists of NVORD groups of cards, each containing:

- ⌈NMORD/4⌉* cards, in (4(2F5.1,5X)) format, containing NMORD pairs of values: the average fragment mass (grains) and the average number of fragments** for each mass category, for one polar zone.
- one card, in (50X,E10.1) format, containing the average fragment initial velocity (fps) for one polar zone.

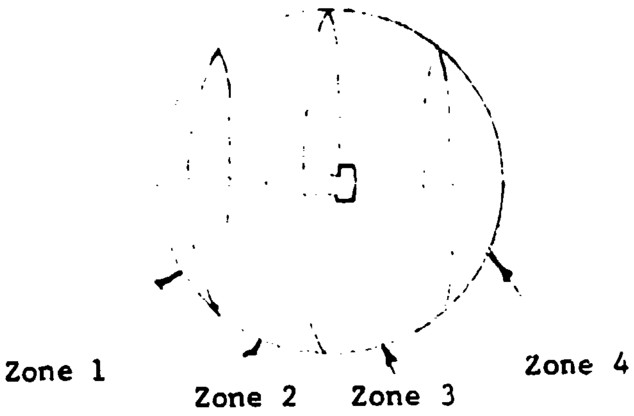
Format 2 Ordnance Data

Data of format 2 is split into two parts.

Part 1 contains NSETS of cards, each consisting of ⌈NMORD/4⌉ groups of NVORD cards. Each card is in (8E10.0) format and contains four pairs of values: the average fragment mass (grains) and the average number of fragments for four mass categories, for one polar zone.

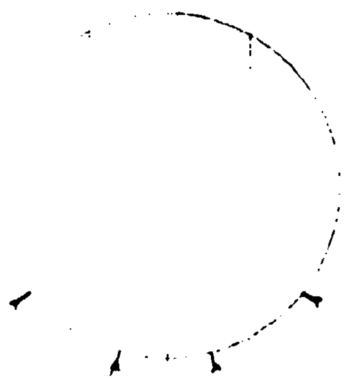
*⌈X⌉ means the smallest integer not less than X.
Examples: ⌈3.14⌉=4, ⌈14⌉=14, ⌈7.00001⌉=⌈7.9999⌉=8.

**Fragment number information in the form of a count of the total number of fragments in a polar zone may be used, provided that the NORM parameter is specified on the $ORDNANCE statement.

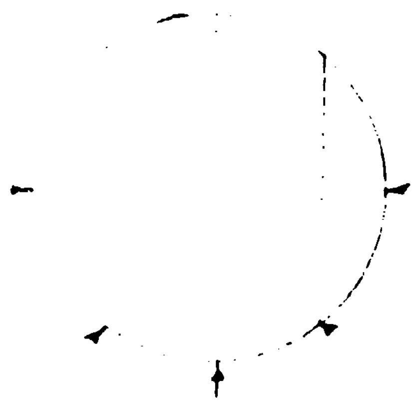


Hypothetical Sphere of Unit Radius Surrounding Ordnance.

Sphere divided into four polar zones, each of equal latitude range.



Measurements Taken at Center of Zones



Measurements Taken at Zone Boundaries (REZONE Parameter Required on $ORDNANCE Statement)

Part 2 contains NSETS sets of cards, each consisting of NVORD cards in (50X,E10.0) format, each of which contains the average fragment initial velocity (fps) for one polar zone.

$BOOM [,FLAGS=α][,NBAR=n][,ORDER=n][,Z=h][,DRNG=r][,NRNG=n][,NEL=n][,NAX=n]

Generate a fragment distribution using ordnance data read according to the prior $ORDNANCE statement. Write the fragment distribution field, with a sentinel record on binary unit 4.

FLAGS=α — α consists of one or two characters. If α is one character, the second flag character is assumed to be zero. If the FLAGS field is omitted, both flag characters are assumed to be zero. The flag characters are used in the sentinel I.D. for the fragment field output.

NBAR=n — Number of barriers placed in the vicinity of the ordnance. If omitted, NBAR=0 is assumed. Barrier description cards, if any, are read after the fragment field title card, NBAR $\leq$ 1.

Z=h — height of target plane (in feet) above ordnance. If Z<0, the ordnance is above the target plane.

DRNG=r — spacing of range points, in feet.

NRNG=n — number of range points on an azimuth ray, NRNG $\leq$ 20.

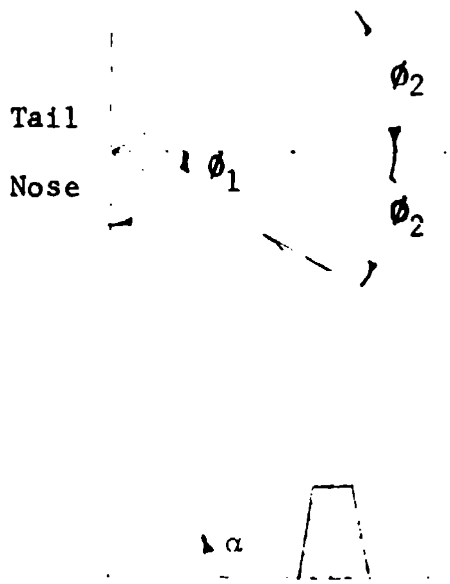
NEL=n — Number of elevation angles to be considered (high register fragments), NEL $\leq$ 18.

NAZ=n — number of azimuth rays in fragment field (a semicircle), NAZ $\leq$ 36.

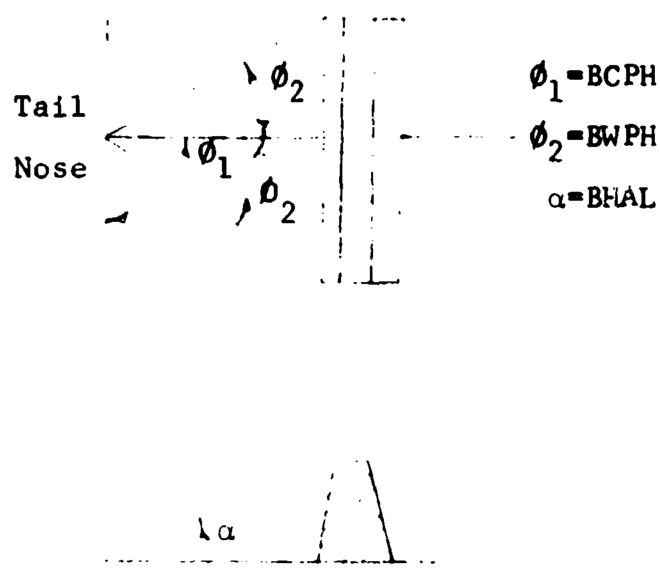
The following cards are read immediately after a $BOOM statement is encountered:

- A title card. Columns 1-72 contain a description of the fragment field.
- If NBAR>0, NBAR barrier description cards are read. The format of each card is:

COLS	FORMAT	NAME	DESCRIPTION
1-5	I5	IBTYP	Barrier type code. 1=barrier is a circular arc, with ordnance at center. 2=barrier is straight and is closest to the ordnance at its midpoint.
5-10			(ignored)
11-20	E10.0	BCPH	Azimuth angle (deg) of barrier midpoint
21-30	E10.0	BWPH	Extent of barrier in azimuth (deg) on either side of midpoint.
31-40	E10.0	BHAL	Angular height of barrier (deg), as seen from the ordnance. If IBTYP=2, this angular height is measured at the midpoint of the barrier.



Circular Barrier



Straight Barrier

```
         ⎧,TABLES                          ⎫
         ⎪,PARAMS                          ⎪
$OUTPUT  ⎨,LIMITS                          ⎬ [,LIST]
         ⎪,EXEC[,PLOT][,KWIKPLOT]          ⎪
         ⎪,RANGE                           ⎪
         ⎩,SOJAC                           ⎭
```

Read function definitions or compute and plot functions of the fragment field variables (velocity, striking angle, and number per square foot, as functions of range and azimuth).

The fragment field is read from binary unit 4, which is presumed to be properly positioned.

TABLES	Read Damage Threshold Velocity Tables or Formula Coefficients*. Between two (2) and five (5) cards are read for each Table or Formula. The formats of these cards are given below. The end of input is marked by a card containing blanks or zeroes in columns 1 and 2.
PARAMS	Read a title card, and fragment threshold and target description cards for each fragment field function. Between two (2) and four(4) cards are read for each function. The formats of these cards are given below. The end of input is marked by a card containing blanks or zeroes in columns 1 and 2.
LIMITS	Read an output parameter card for each fragment field function. The format of this card is given below. The end of input is marked by a card containing blanks or zeroes in columns 1 and 2.

*The formula used is the empirical relationship derived under project THOR.

EXEC

Compute and plot specified functions of a previously computed fragment field. Functions are specified on the data card following this control card. The format of this card is given below.

The fragment field is read from binary unit 4, which is presumed to be properly positioned.

The functions are specified using two digit codes, whose meanings are:

01 total number of fragments (per sq ft)

02 total fragment mass (lb)

03 total fragment momentum (lb-sec/ft^2)

04 total fragment kinetic energy (ft-lb/ft^2)

05-07 (unused)

08* average fragment impact velocity (ft/sec)

09* average fragment impact angle (radians)

1j number of damaging fragments, using table j.

2j probability of damage, using table j.

3j compute and plot 1j and 2j together (this is cheaper than specifying 1j and 2j separately)

4j number of damaging fragments, using Thor coefficient set j.

5j probability of damage, using Thor coefficient set j.

6j compute plot 4j and 5j together (this is cheaper than specifying 4j and 5j separately)

7j-9j (unused)

Up to eight (8) functions may be computed and plotted at one time. To plot more than eight functions, the fragment field file must be repositioned, and another set of plots requested.

*Functions 08 and 09 must both be specified if either is, and must be the last function codes specified.

Note that a function code of 3j or 6j causes two plots to be generated, and that specification of functions 08 and 09 counts for 3 functions, even though only two plots are generated.

PLOT	cause function values to be written to the plot file (file 2) for later processing by a contour plotting program.
KWIKPLOT	generate contour plots of functions values on the printer.
RANGE	read plot range parameters from data card following this control card. The format of this card is given below.
SOJAC	read characters to be used for printer plotter (see KWIKPLOT) from data card following this control card. The format of this card is given below.
LIST	list data read with this control card.

Data cards read after a $OUTPUT,TABLES statement. Between two and five cards are read for each table or formula coefficient group.

COLS	FORMAT	NAME	DESCRIPTION
Heading Card			
1	I1	IT	type code. If IT=1,2 or 3, table IS is to be read. If IT=4,5 or 6, coefficient group IS is to be read. IF IT=0 and IS=0, the end of this group of data cards has been reached.
2	I1	IS	table or coefficient group number. $0 \le IS \le 9$.
3-5			(ignored)
6-10	I5	NTB	(used only for tables) number of M-V pairs to be read. NTB≤14.
11-15	I5	JDUM	(used only for tables) interpolation mode. All table values are read in true form, but linear interpolation of the table during processing may require either a log or linear scale, according to JDUM: =00 M linear, V linear =01 M linear, V log =10 M log, V linear =11 M log, V log

Coefficients (formula coefficients only)

1-70	7E10.0	(COEF(I),I=1,7)	- Thor formula coefficients, group IS.

Fragment Mass Values (tables only)

NTB masses are read, 7 per card.

1-70	7E10.0	(XTB(I),I=1,7)	- Fragment mass values, (grains) for table IS.

Damage Threshold Velocity Values (tables only)

NTB velocities are read, 7 per card.

1-70	7E10.0	(YTB(I),I=1,7)	- Damage threshold velocities (ft/sec) for table IS.

Data cards read after a $OUTPUT,PARAMS statement. Between two and four cards are read for each fragment field function.

COLS.	FORMAT	NAME	DESCRIPTION
<u>Heading Card</u>			
1	I1	IT	Function type. If IT=0 and IS=0, the end of this group of data cards has been reached. If IT=0 and IS>0, then the function requires neither a table nor a Thor formula, and the function type is given by IS. IF IT>0, then IT determines the function type. =1 number of damaging fragments per sq ft, using table IS. =2 probability of damage, using table IS. =4 number of damaging fragments per sq ft using coefficient set IS in the Thor formula. =5 probability of damage using coefficient set IS in the Thor formula.
2	I1	IS	table number (if IT=1 or 2), coefficient group number (if IT=4 or 5), or function type (if IT=0): =1 total number of fragments per sq ft =2 total fragment mass (lb) per sq ft =3 total fragment momentum (lb-sec) per sq ft =4 total fragment kinetic energy (ft-lb) per sq ft =8 average fragment impact velocity (ft/sec) =9 average fragment impact angle (radians)
3			(ignored)
4	A1	IVP	Function variant code. A blank is treated as a 'zero' character. IVP is used to distinguish plots of the same function, but with different parameters.
5			(ignored)
6-10	I5	NF	Number of parameters $(0 \leq NF \leq 14)$ If $NF < 14$, parameters not read are set to zero.

COLS.	FORMAT	NAME	DESCRIPTION

Title Card

1-72	12A6	(ITTLF(I),I=1,12) - function title, which will appear on all plots of this function.

Parameter Card(s)

NF parameters are read, 7 per card. If NF=0, no cards are read.

1-70	7E10.0	(F(I),I=1,NF) - parameters affecting function performance.

Parameters 1 through 7 have the same meaning for all functions:

F(1) miminum fragment weight (grams). All lighter fragments are ignored.

F(2) minimum fragment velocity (ft/sec). All slower fragments are ignored.

F(3) minimum fragment momentum (lb-sec). All fragments whose momentum is too small are ignored.

F(4) minimum fragment kinetic energy (ft-lb). All fragments whose kinetic energy is too small are ignored.

F(5) - F(7) (reserved)

Parameters 8 through 14 are used only if IT>0.

F(8) target plan area (ft^2)

F(9) average target elevation area (ft^2)

F(10) - F(11) are used only if IT=4 or 5.

F(10) target thickness (in.)

F(11) target hardness, where applicable.

F(12) - F(14) (reserved)

Data cards read after a $OUTPUT,LIMITS statement. One card is read for each function.

COLS.	FORMAT	NAME	DESCRIPTION
1	I1	IT	Function code (see PARAMS discussion above)
2	I2	IS	subcode or table code (see PARAMS discussion above) Note that if IT=0 and IS=0 the end of this group of data cards has been reached.
3-10			(unused)
11-14	I4	ISM	Plot smoothing code. IF ISM=0, do no smoothing. IF ISM=1 to 5, smooth by replacing each value by the average of the values in a 2*ISM+1 raster square centered on that value.
15	I1	ISC	Plot scale code. ISC=0 to request a linear plot. ISC=1 to request a plot of the log of the function
16-20			(unused)
21-30	E10.0	OZMIN	minimum value to be plotted.
31-40	E10.0	OZMAX	maximum value to be plotted. If OZMIN>OZMAX, the plot range is set internally.

COLS.	FORMAT	NAME	DESCRIPTION

Data card read after $OUTPUT,EXEC statement. See discussion of the EXEC parameter for the list of valid function codes.

COLS.	FORMAT	NAME	DESCRIPTION
1-2	I2		first function code
3-5			(ignored)

Subsequent function codes go in columns 6-7, 11-12, etc.

Data card read after $OUTPUT,SOJAC statement.

COLS.	FORMAT	NAME	DESCRIPTION
1-5			(ignored)
6-10	I5	NZ	number of function value ranges to be plotted on printer. NZ≤50
11-19			(ignored, leave blank)
20-69	50A1	(CH(I),I=1,NZ)	- plot characters for each function value range.

Data card read after $OUTPUT,RANGE statement.

COLS.	FORMAT	NAME	DESCRIPTION
1-5			(ignored)
6-10	I5	NX	number of resolution increments in X direction (tail to nose). NX≤81.
11-15	I5	NY	number of resolution increments in Y direction (laterally from waist of munition). Normally NY=(NX-1)/2+1. NX≤41.
16-20			(unused)
21-30	E10.0	XMIN	minimum X value (ft)
31-40	E10.0	YMIN	minimum Y value (ft). Normally YMIN=0.
41-50	E10.0	DX	X increment (ft). Normally NX,XMIN and DX are set so that XMAX=-XMIN where XMAX is XMIN+DX*(NX-1).
51-60	E10.0	DY	Y increment (ft). Normally NY and DY are set so that YMAX=XMAX where YMAX is DY*(NY-1) if YMIN=0.

Sample Data Deck

the deck listed on the next page is set up to do the following job:

1) Read munition data for munition No. 4 (mod 1). Gaps in the data are to be filled in, and the data is to be listed neatly in tabular form.

2) A fragment field is to be computed, on a polar grid consisting of 8 ranges spaced 250 feet apart, and 18 azimuth rays, spread evenly over a half-circle (at azimuths 5, 15, ..., 175 deg). After the fragment field has been written onto tape 4, an end of file is written and the tape is rewound.

3) Output specifications are read. These include

   a. the printer-plotter character set,

   b. the size and resolution of the plot,

   c. the trivial (0-velocity) fragment threshold velocity table,

   d. the function limits and scale (all are to be plotted in log units), and

   e. the function titles and parameters.

4) Plots are generated. The function specifications 01 and 30 cause functions 01, 10 and 20 to be generated and plotted.

```
$PRINT CONTROL,DATA
$ORDNANCE FORMAT=2,NORM,SMOOTH,LIST
MUNITION 04 MOD 1
   16   36    4-1   660.
```

(The munition data goes here.)

```
$BOOM NAZ=18,DRNG=250.,NRNG=8     $    2000 FOOT RADIUS, 10 DEG AZ SECTORS
GROUND LEVEL FRAGMENT FIELD
$TIME
$TAPE 4,WEOF
$TAPE 4,REWIND
$OUTPUT SQUAC LIST
         9          -123456789+
$OUTPUT RANGE LIST
        81    41       -2000.      0.          50.         50.
$OUTPUT TABLES LIST
10         2    0
0.          100000.
0.          0.
  END
$OUTPUT LIMITS LIST
01               1        .00016667 .16667
10               1        .00016667 .16667
20               1        .0001      .1
  END
$OUTPUT PARAMETERS LIST
01         4
AREAL DENSITY OF ALL FRAGMENTS (TARGET IS SPHERE OF UNIT CROSS-SECTION)
0.         0.          0.          0.
10         9
AREAL DENSITY OF DAMAGING FRAGMENTS (K.E. GT 58 FT-LB)
0.         0.          0.          58.
1.33       9.0
20         9
PROBABILITY OF HIT BY DAMAGING FRAGMENT (K.E. GT 58 FT-LB)
0.         0.          0.          58.
1.33       9.0
  END
$OUTPUT EXEC LIST KWIKPLOT
01   30
$TIME
$STOP
```

Sentinel Records

The munition data file (if one is used), the fragment field file, and the output function file (if one is used) each consist of several groups of data. Each group is preceded by a sentinel record which identifies the group, as described in the following section. In addition a special sentinel record is written after the last data group.

A sentinel record exists in two forms, one for formatted files (the BCD munition data file), and one for unformatted files (the binary fragment field file and the binary output function file).

Formatted sentinel record:

char. 1-6	'*HEAD*'
char. 7-14	an 8-character search key. (If fewer than 8 characters, blanks are added.)

Search for a formatted data group using the statement:

```
$TAPE n,BCD,SRCH=key,BKSP
```

Unformatted sentinel record:

word 1*	'*HEAD*'
word 2**	an 8-character search key. (If fewer than 8 characters, blanks are added.)

Search for an unformatted data group using the statement:

```
$TAPE n,SRCH=key,BKSP
```

*Two words are needed when using a computer with four characters per word.

**Two words are needed when using a computer with four or six characters per word.

Sentinel Record Keys

1. A 3-character key of the form

   XXI

   is used for munition data. (BCD tape)

   XX = munition ID number. $00 \leq XX \leq 99$

   I = a modification code. Normally I=0.

2. A 5-character key of the form

   XXIJK

   is used for fragment field tables. (Binary tape 4)

   XX,I as above

   J,K = fragment field modification code to reflect changes in resolution, target plane altitude, barrier placement, etc.). Normally J=K=0.

3. An 8-character key of the form

   XXIJKYYL

   is used for output function tables. (Binary tape 2)

   XX,I,J,K as above

   YY = output function code. (See $OUTPUT discussion.)

   L = output function modification code(to reflect changes in threshold or target parameters, etc.). Normally L=0.

4. The sentinel record which should end a file has an 8-character key which is

   99999999

   A sentinel record may be written with the statement

   $TAPE n[,BCD],WEOF

Timing

The generation of the fragment field is the most time consuming part of the program. Several parameters affect timing in this phase.

NM	number of mass categories into which munition data is divided
NAZ	number of azimuth angles
NEL	number of elevation angles
DRNG	range increment (ft)

All operations are performed for each mass category at each azimuth. Low-register fragment trajectories are computed for ranges R_i=DRNG*i, out to $R_i=R_{max}$, which corresponds to an initial elevation angle of α_{max}. High-register fragment trajectories are computed for elevations $\alpha_i=(i-1/2)\Delta\alpha$, where $\Delta\alpha=90°/NEL$, and where $\alpha_{max}<\alpha_i<90°$ holds.

Thus the computation time for low register fragment trajectories increases as DRNG decreases, and the computation time for high register fragment trajectories increases as NEL increases. So the expected computation time is given by

T=A*NM*NAZ*(1+B/DRNG+C*NEL)

Because all runs done for this project used the same values of NAZ, DRNG and NEL, the constants A, B and C cannot be determined at this time. However, the approximate relation (in sec)

T=7 x NM

and has been observed on the Univac 1108, with NAX=18, DRNG=250 and NEL=18.

APPENDIX B

CONTOURS OF FRAGMENT NUMBER DENSITIES
AND INJURY PROBABILITIES

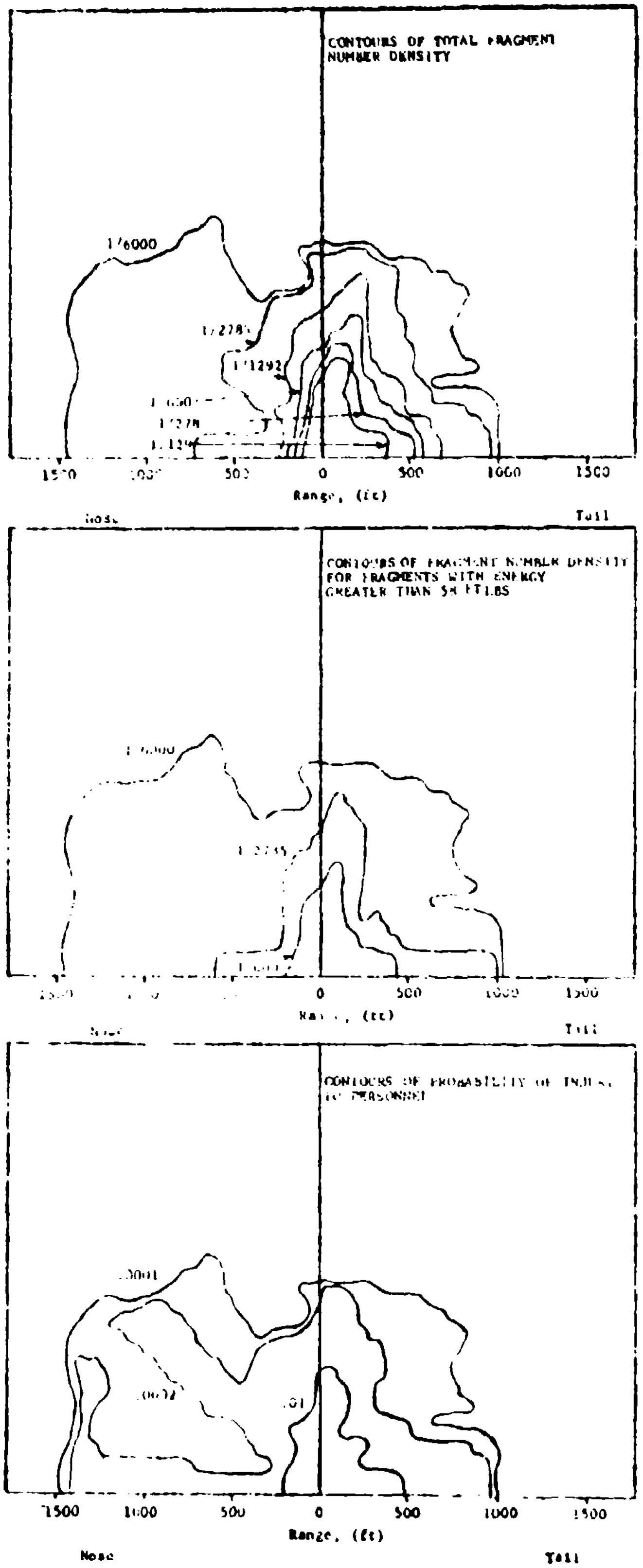


Fig. B1 500 LB LOW DRAG BOMB MARK 82 MOD 1
(H-6 LOAD)

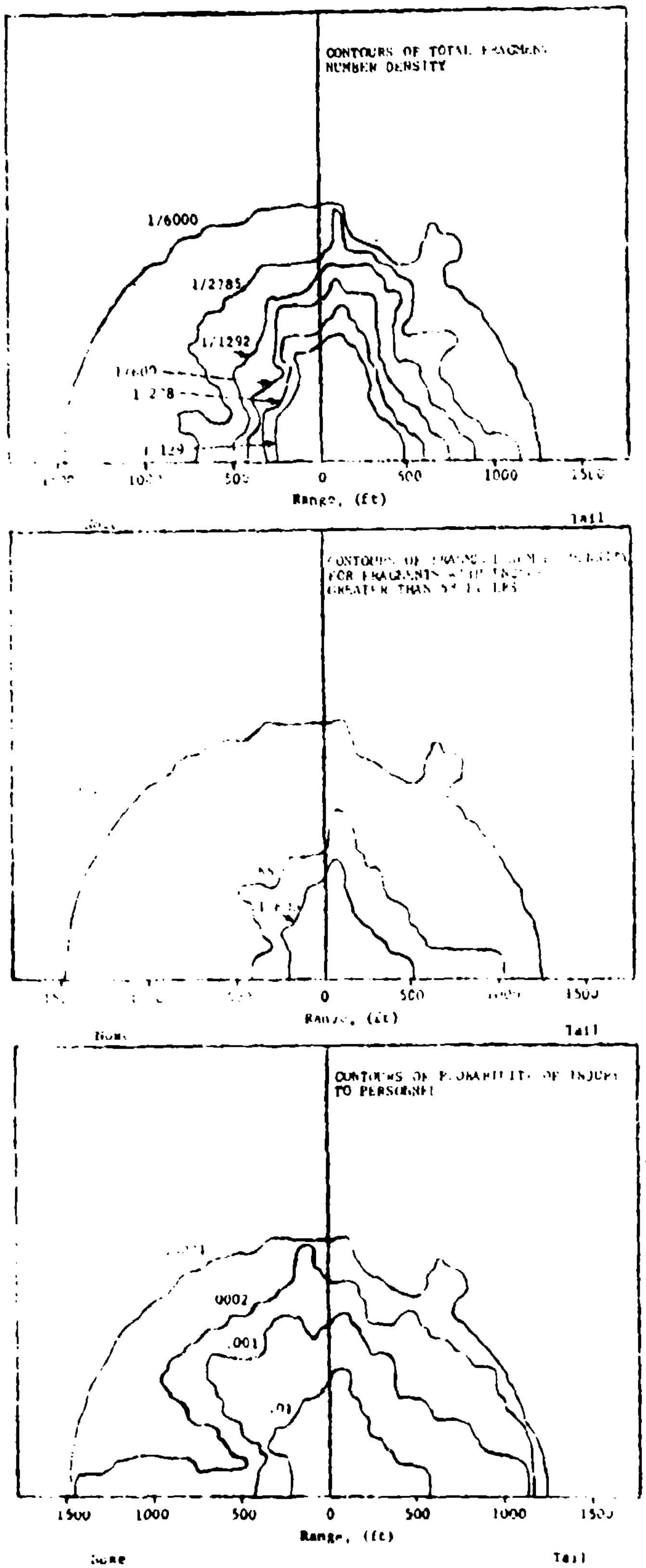
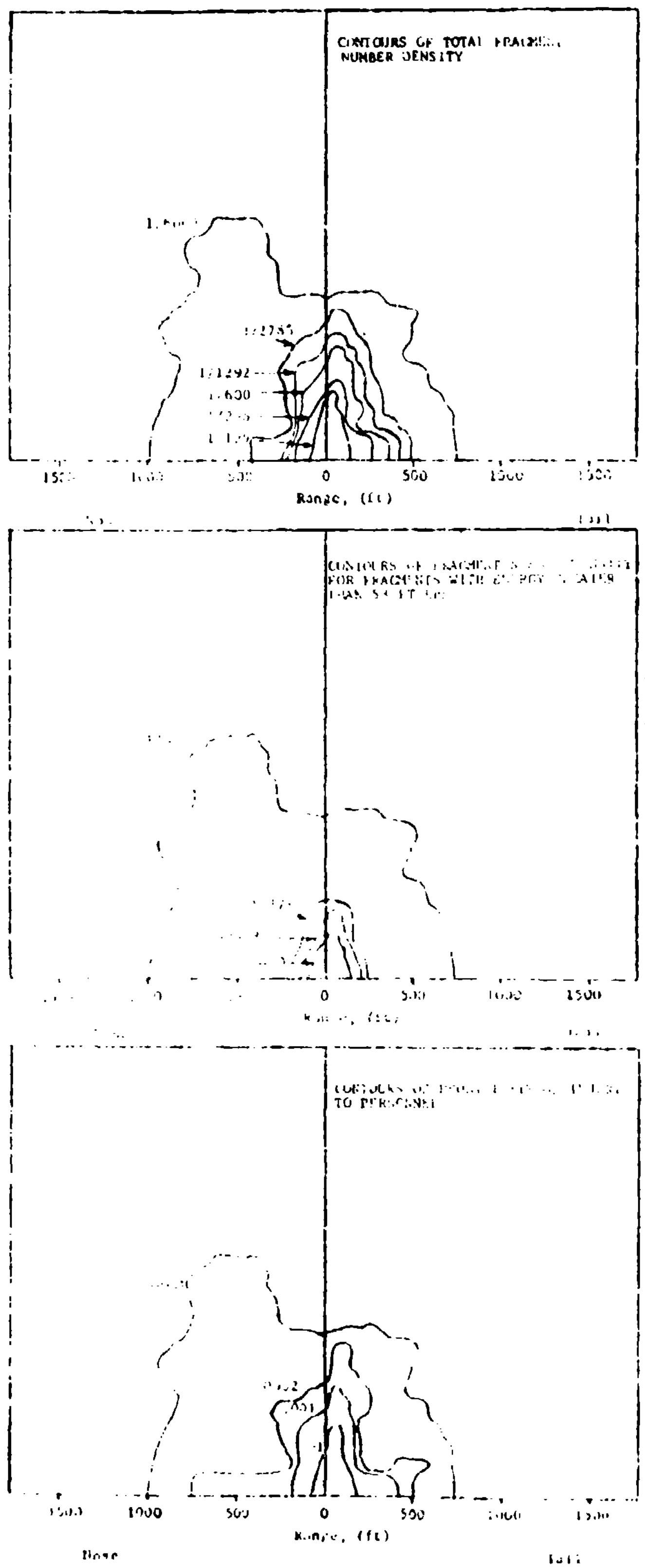


Fig. B2 750 LB BOMB M117A2
(TRITIONAL LOAD)



F . B3 105 MM HOWITZER SHELL M1
(COMP. B LOAD)

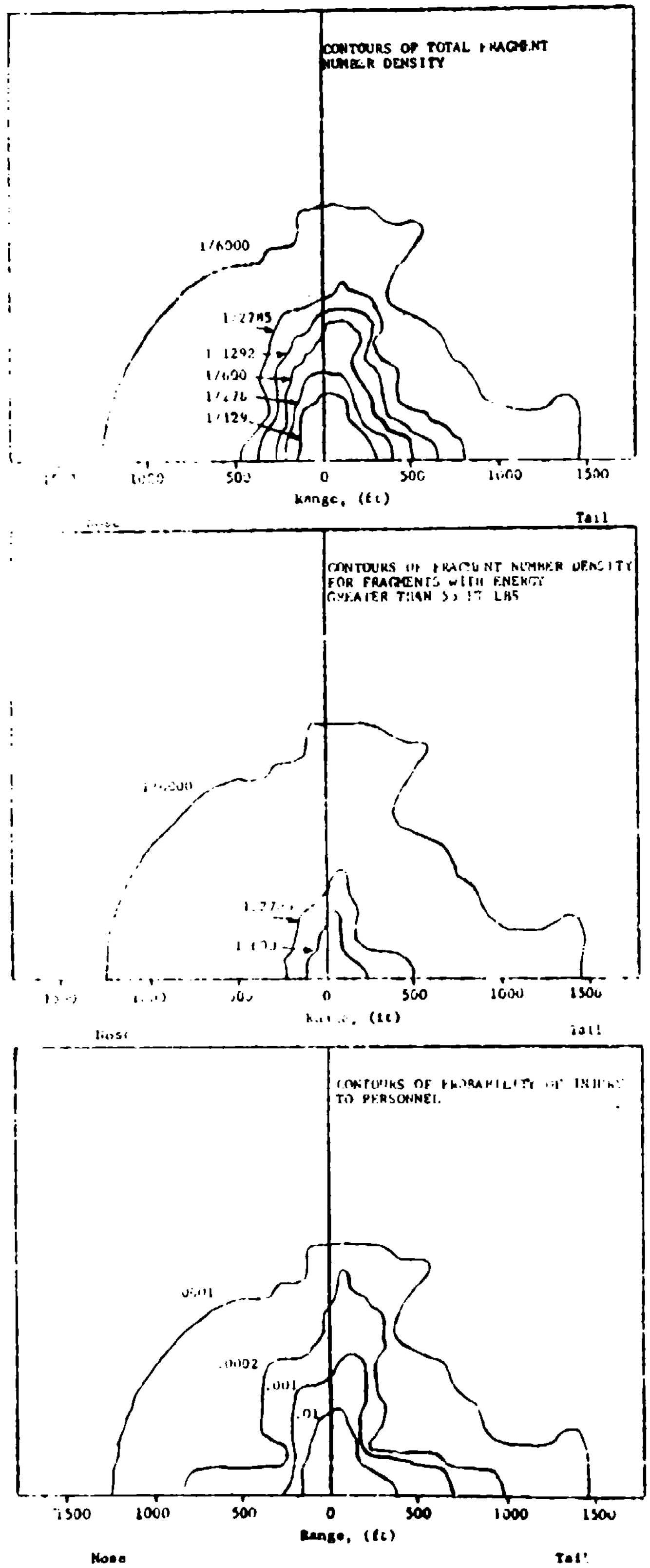


Fig. B4 155 MM HOWITZER SHELL M107
(COMP. B LOAD)

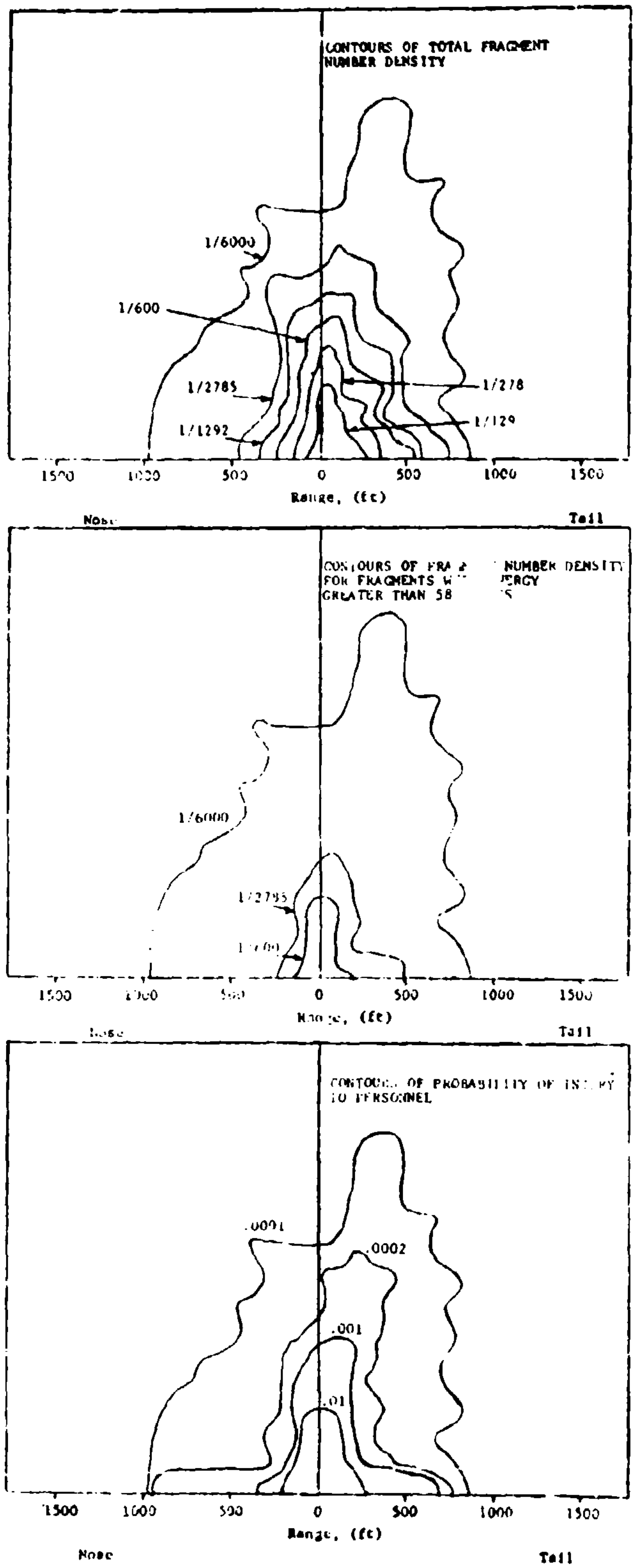


Fig. B5 175 MM GUN SHELL M437A2
(COMP. B LOAD)

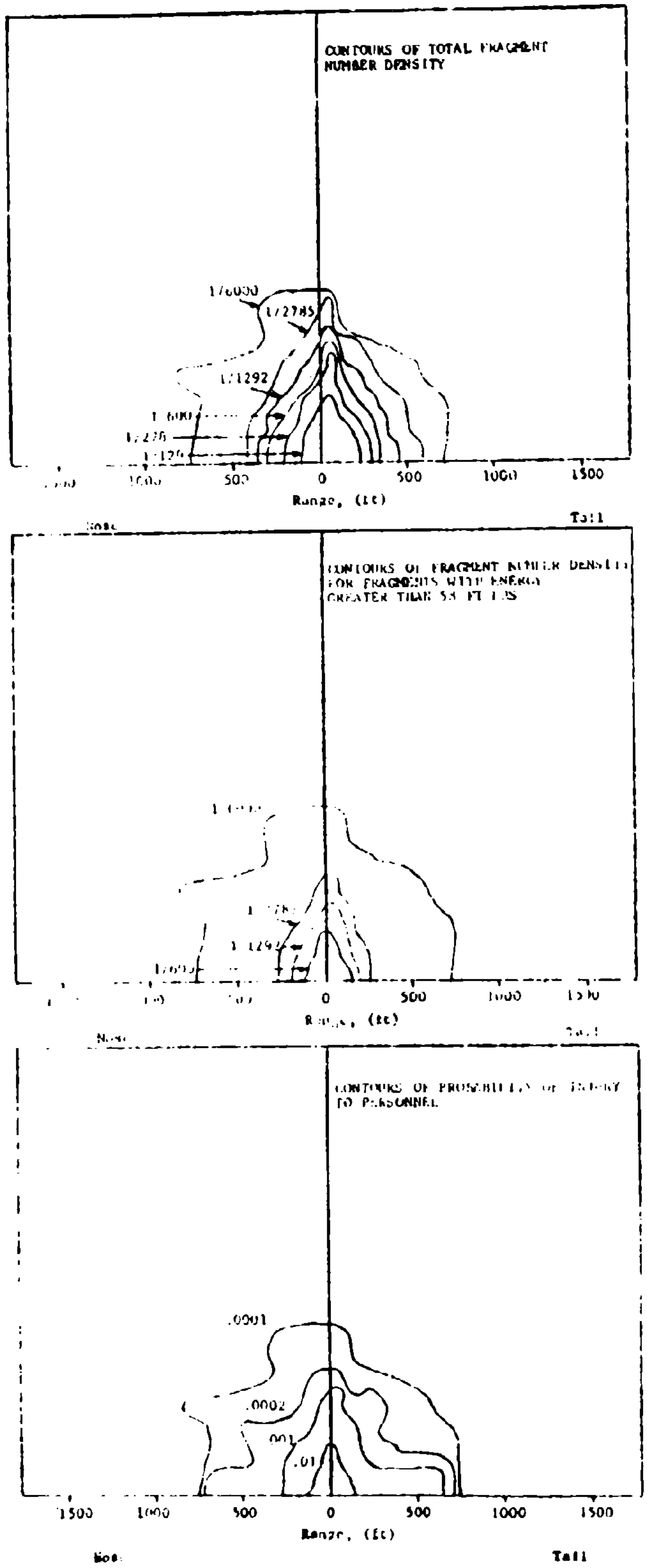


Fig. B6 5"/38 PROJECTILE MARK 49
(COMP. A3 LOAD)

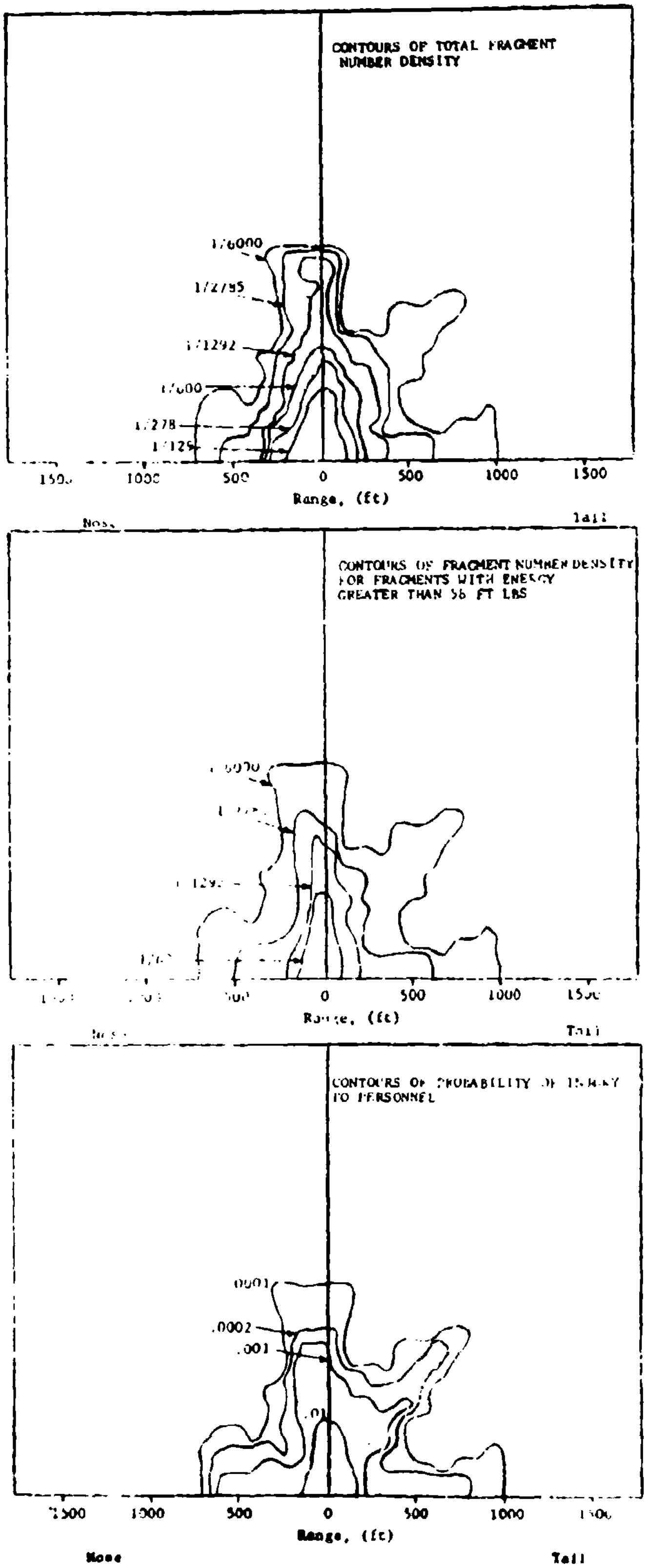
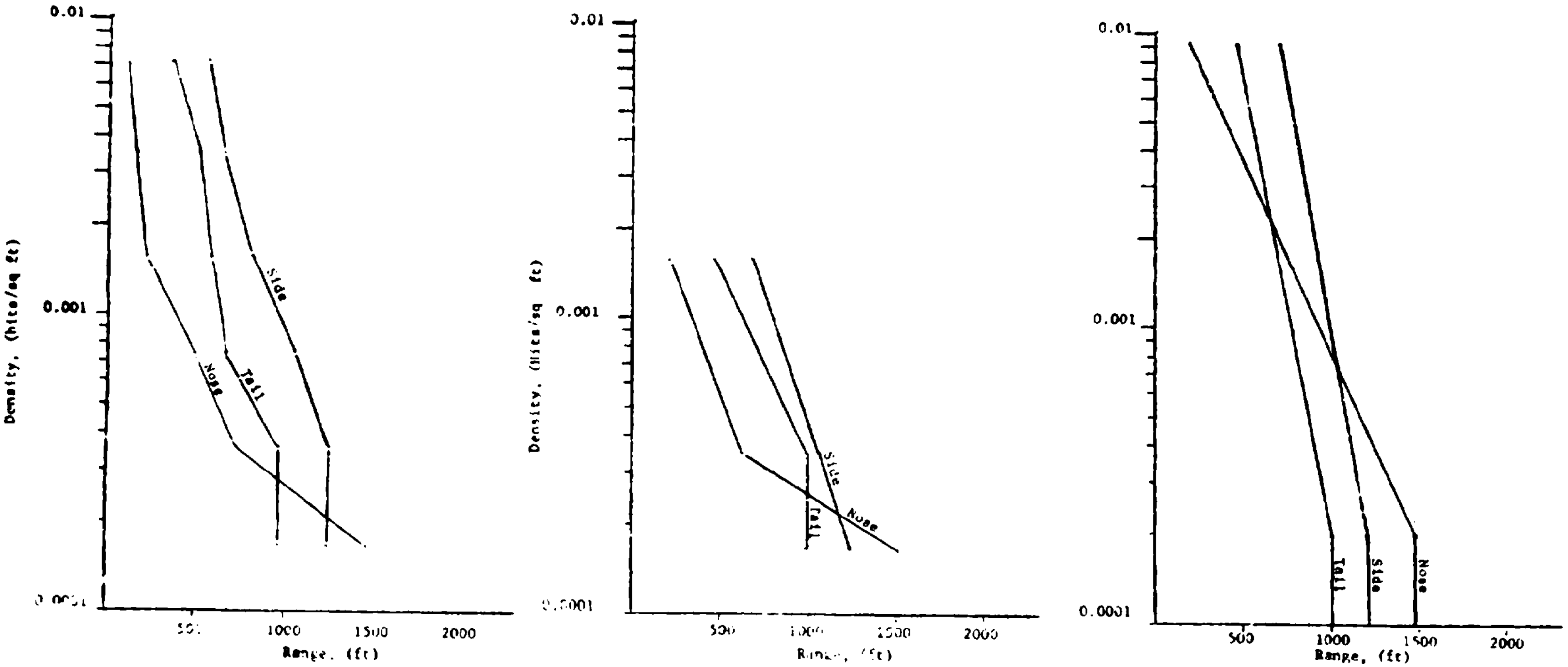


Fig. B7 8"/55 PROJECTILE MARK 25
(EXPLOSIVE D LOAD)

APPENDIX C

AVERAGE CURVES OF FRAGMENT NUMBER DENSITIES AND INJURY PROBABILITIES

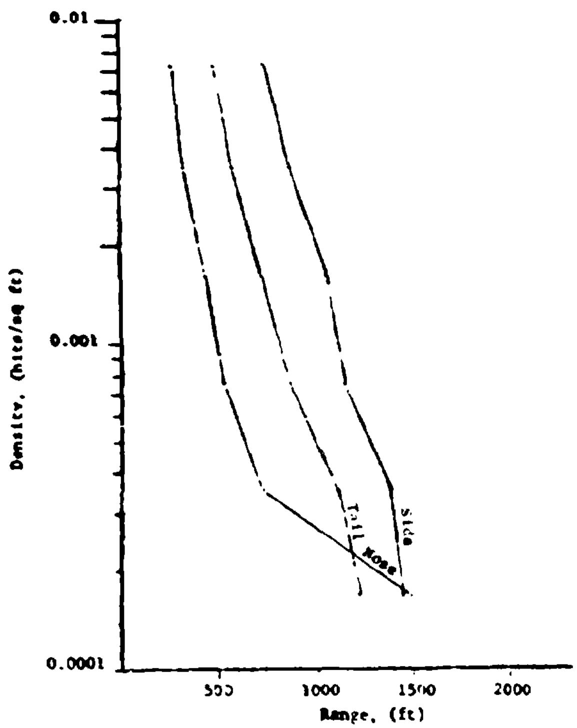


A. TOTAL FRAGMENT NUMBER DENSITY

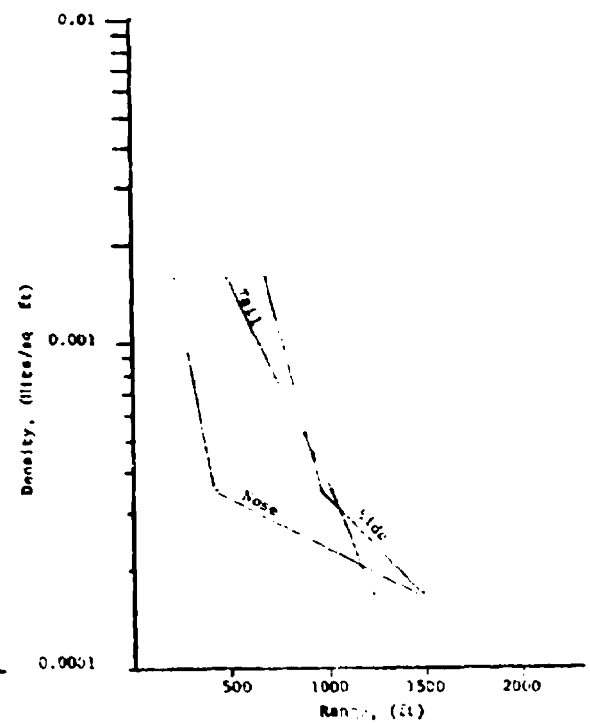
B. FRAGMENT DENSITY FOR FRAGMENTS GREATER THAN 5 ... [illegible]

C. PERSONNEL FRAGMENT INJURY PROBABILITIES

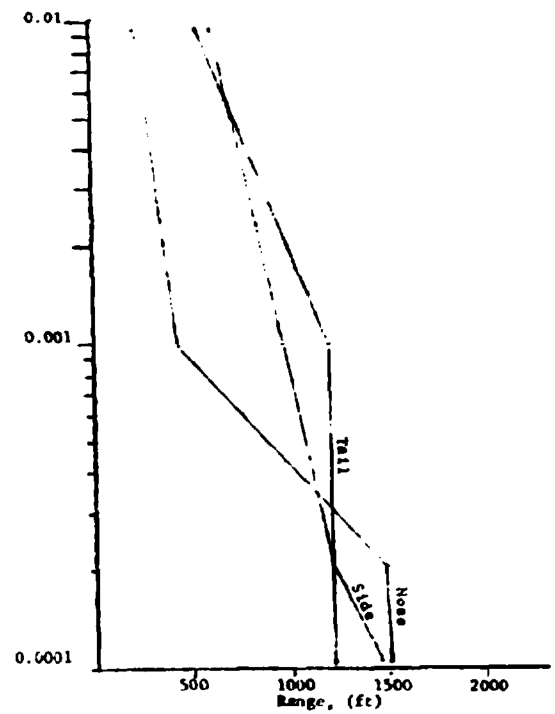
Fig. C1 500 LB LOW DRAG BOMB MARK 82 MOD 1 (H-6 LOAD)



A. TOTAL FRAGMENT NUMBER DENSITY

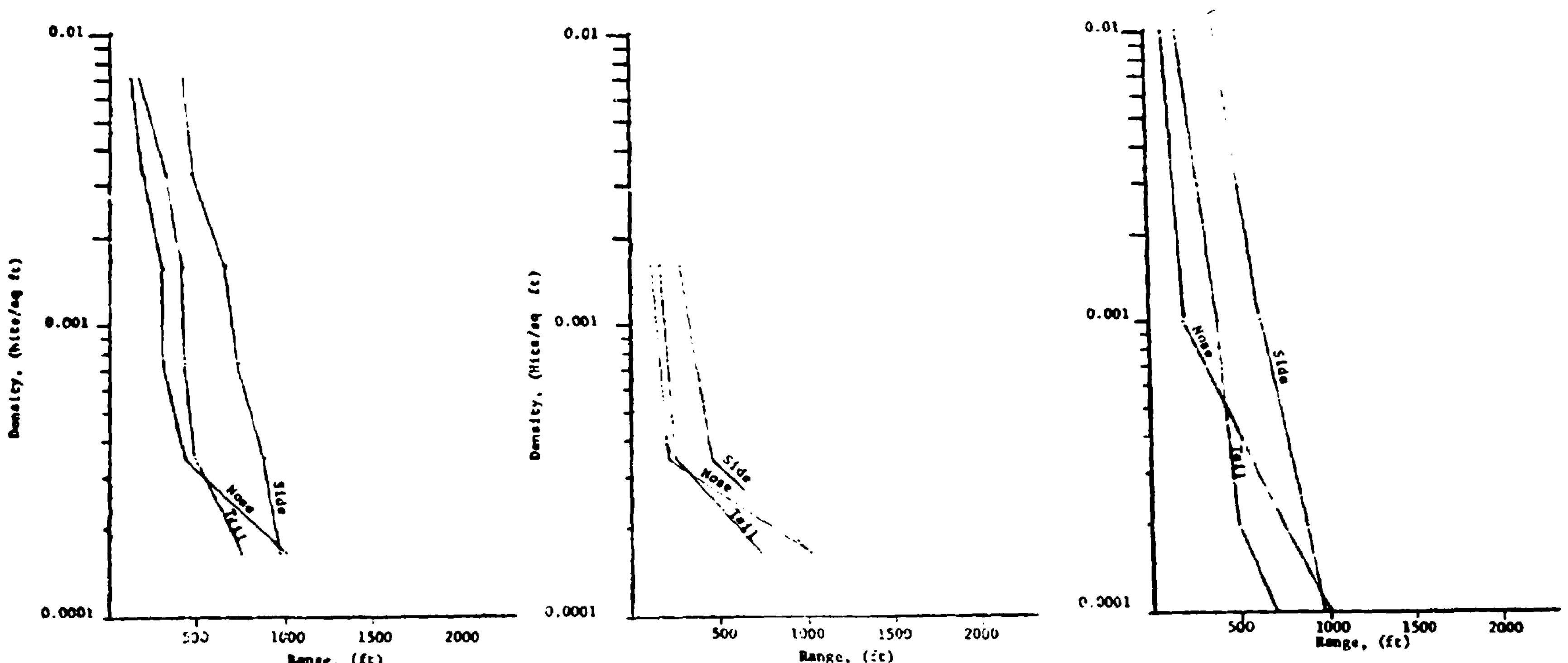


B. FRAGMENT DENSITY FOR FRAGMENTS GREATER THAN [illegible]



C. PERSONNEL FRAGMENT INJURY PROBABILITIES

Fig. C2 750 LB BOMB M117A2 (TRITIONAL LOAD)

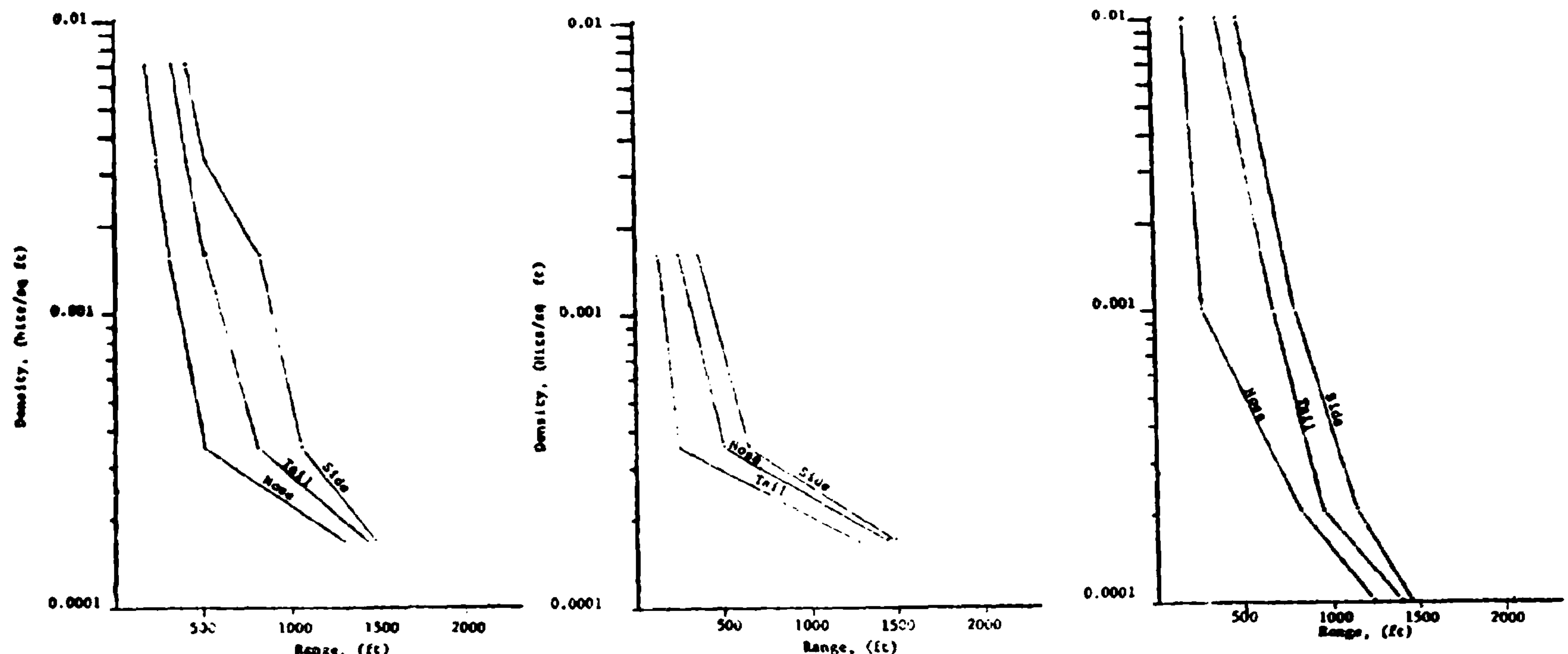


A. TOTAL FRAGMENT NUMBER DENSITY

B. FRAGMENT DENSITY FOR FRAGMENTS GREATER THAN 58 FT LBS

C. PERSONNEL FRAGMENT INJURY PROBABILITIES

Fig. C3 105 MM HOWITZER SHELL M1 (COMP. B LOAD)

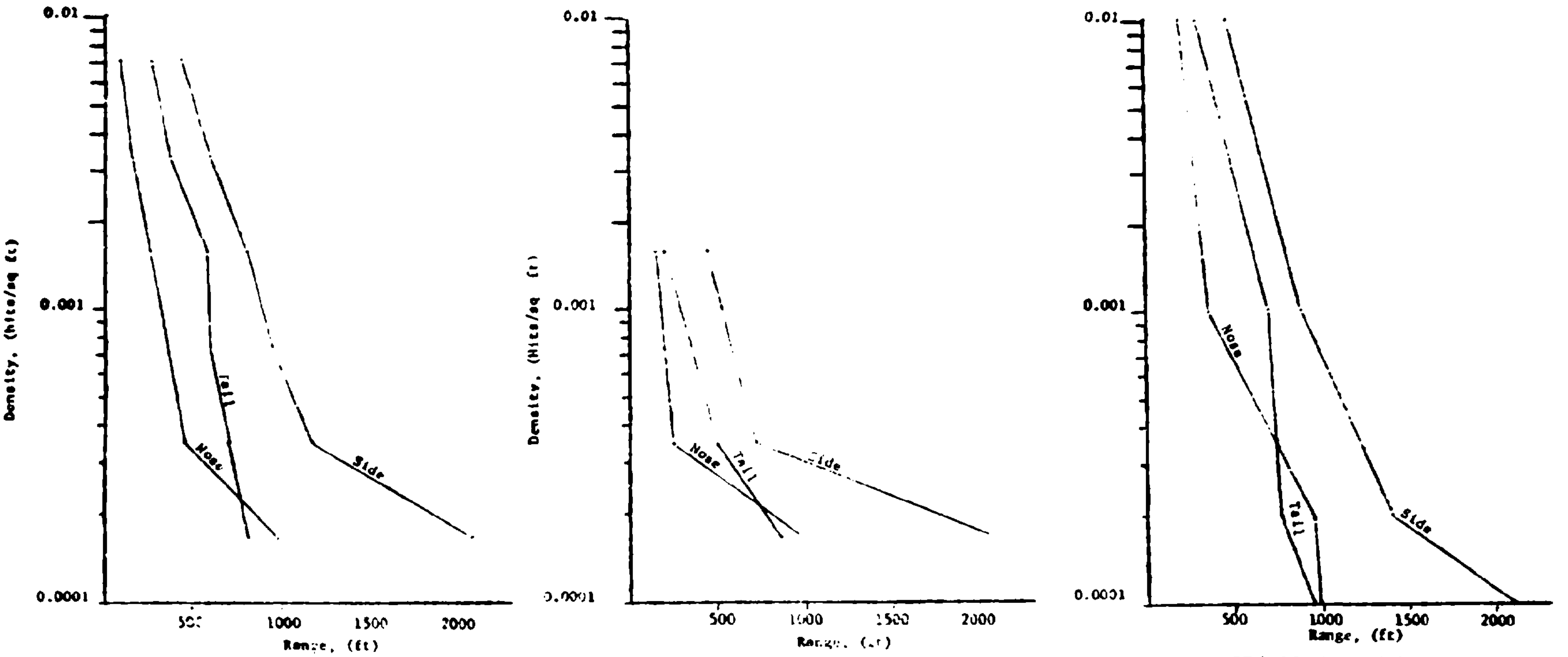


A. TOTAL FRAGMENT NUMBER DENSITY

B. FRAGMENT DENSITY FOR FRAGMENTS GREATER THAN 58 FT LBS

C. PERSONNEL FRAGMENT INJURY PROBABILITIES

Fig. C4 155 MM HOWITZER SHELL M107 (COMP. B LOAD)

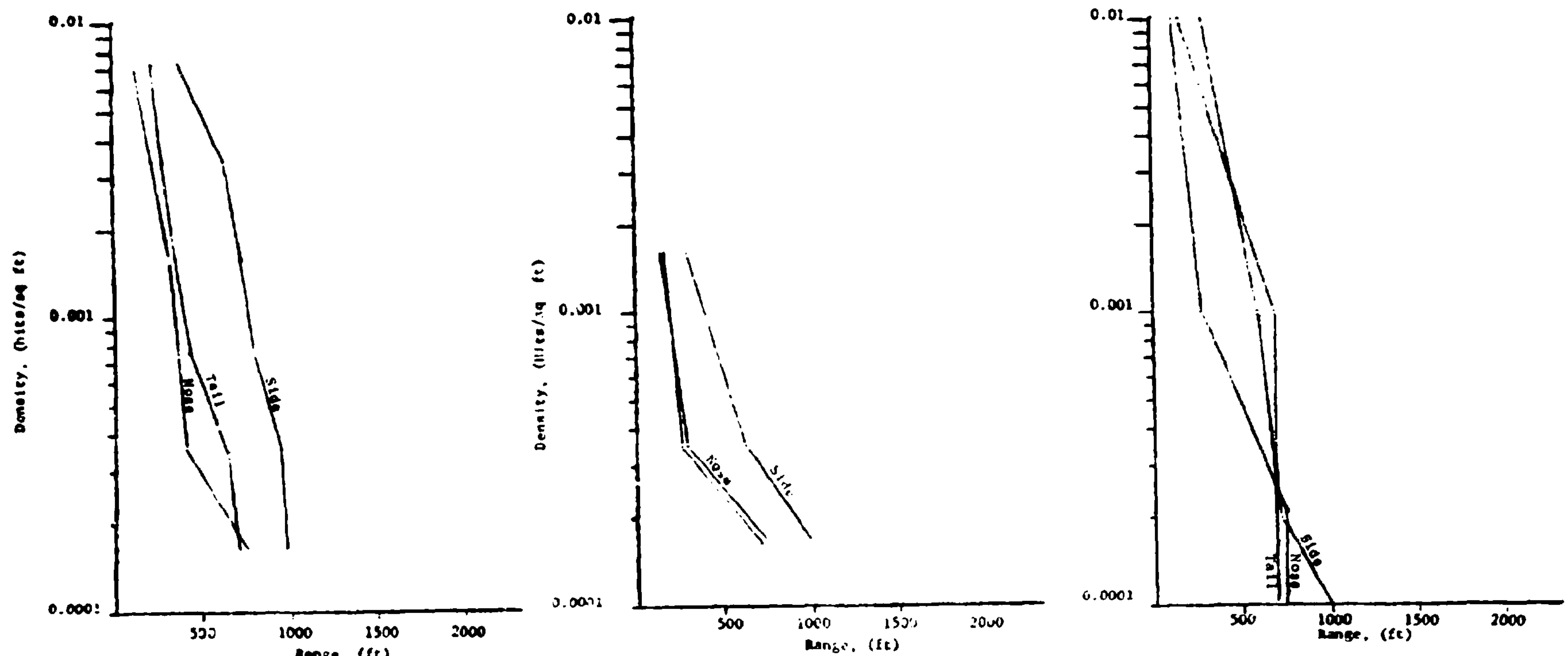


A. TOTAL FRAGMENT NUMBER DENSITY

B. FRAGMENT DENSITY FOR FRAGMENTS GREATER THAN [illegible] LBS

C. PERSONNEL FRAGMENT INJURY PROBABILITIES

Fig. C5 175 MM GUN SHELL M437A2 (COMP. B LOAD)

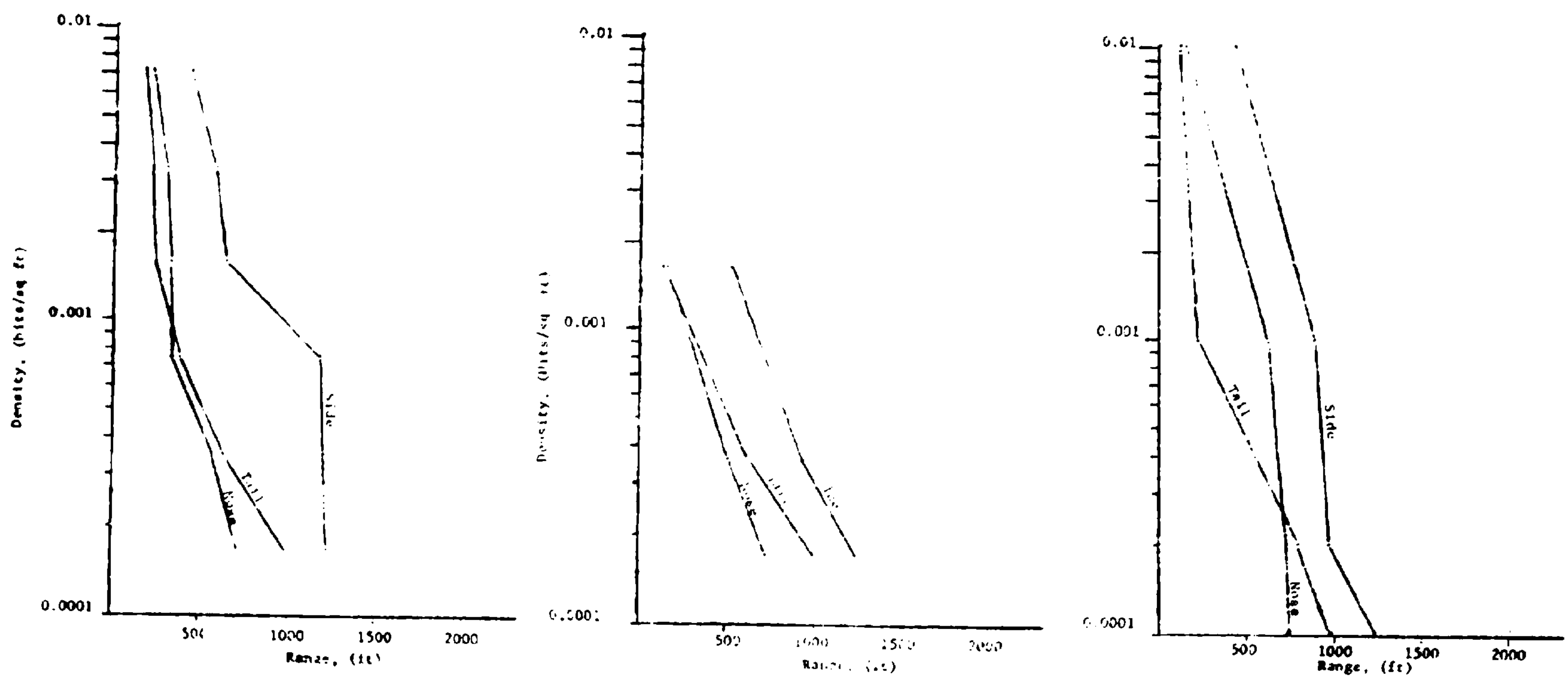


A. TOTAL FRAGMENT NUMBER DENSITY

B. FRAGMENT DENSITY FOR FRAGMENTS GREATER THAN 58 FT LBS

C. PERSONNEL FRAGMENT INJURY PROBABILITIES

Fig. C6 8"/55 PROJECTILE MARK 25 (EXPLOSIVE D LOAD)



A. TOTAL FRAGMENT [illegible] DENSITY

B. FRAGMENT DENSITY FOR FRAGMENTS GREATER THAN 5 [illegible]

C. PERSONNEL FRAGMENT INJURY PROBABILITIES

Fig. C7 8"/55 PROJECTILE MARK 25 (EXPLOSIVE D LOAD)

APPENDIX D

COMPUTER PROGRAM LISTINGS

```
C     GIVEN STD TRAJECTORY TABLES (TAPE1) AND ORDNANCE DATA (INPUT OF    TRAG0010
C     BCD TAPE), GENERATE TABLE OF FRAGMENT INFO. ON SEMI-CIRCULAR GRID.TRAG0020
C     STORE FRAGMENT FIELD TABLES ON TAPE 4, IF REQUESTED.               TRAG0030
C     THEN COMPUTE OPTIONAL FUNCTIONS OF FRAGMENT INFO.  OUTPUT FUNCTIONTRAG0040
C     I.D. CODES AND VALUES (TAPE2)                                      TRAG0050
C                                                                        TRAG0060
C  CONTROL CARD TABLES AND GENERAL PROGRAM CONTROL VARIABLES             TRAG0070
      COMMON /CNTRL/ GCCRD,GCCPRT,KTCD,FCC(8),KARD(14),PC,PS,PD,PP,      TRAG0080
     * TXTIME(2),TXDATE(2),WHY(20),KKARD(800)                            TRAG0090
      LOGICAL GCCRD,GCCPRT,WHY                                           TRAG0100
      INTEGER TXTIME,TXDATE,FCC,PC,PS,PD,PP                              TRAG0110
C                                                                        TRAG0120
      EQUIVALENCE (KKARD(12),KTYPE)                                      TRAG0130
C                                                                        TRAG0140
      LOGICAL PRTCCD                                                     TRAG0150
      EQUIVALENCE (KKARD(497),PRTCCD)                                    TRAG0160
C                                                                        TRAG0170
C  SYNTAX TABLES                                                         TRAG0180
      INTEGER XSYN(64),TSYN(64),SSYN(4),SYND,SYNE                        TRAG0190
      LOGICAL SYNTAX                                                     TRAG0200
      EXTERNAL CCARD                                                     TRAG0210
C***  SYND IS A MARKING CHARACTER TO POINT AT SYNTAX ERRORS (11-7-8 PCH)TRAG0220
      DATA NSSYN,SYND,SYNB/4,1H*,1H /,                                   TRAG0230
     * TSYN/5*0,4,30*0,5,5,0,2,3*0,6,0,0,3,11*0,6,5*0/                   TRAG0240
C***  STD CONTROL-CARD SYNTAX RULES, MODIFIED FOR $END OF STATEMENT      TRAG0250
C***  KKARD(4)=2 (NEW STMNT ON SAME CARD) SHOULD THEN NEVER APPEAR       TRAG0260
      DATA XSYN/                                                         TRAG0270
     * 00202000200,00000000020000, 00202000400,00000000020011,           TRAG0280
     * 00202000200,00000000050000, 00202000200,00000000070000,           TRAG0290
     * 00202000400,00000000050006, 00202000300,00000000160000,           TRAG0300
     * 00202000400,00000000070010, 00202000300,00000000260000,           TRAG0310
     * 00202000000,00001000140012, 00202000200,00000000000013,           TRAG0320
     * 00202000100,00000000000000, 00202000400,00000000140015,           TRAG0330
     * 00202000300,00000000160022, 00202000400,00000000160017,           TRAG0340
     * 00204000102,00000000200024, 00701000403,00000000000000,           TRAG0350
     * 00701000402,00000000000000, 00202000200,00006000000023,           TRAG0360
     * 00202000100,00006000000024, 00202000000,00002000330025,           TRAG0370
     * 00202000600,00001000260000, 00202000100,00000000270030,           TRAG0380
     * 00701000404,00000000000000, 00202000600,00000000310032,           TRAG0390
     * 00202000600,00007000260014, 00201000000,00007000260000,           TRAG0400
```

```
     * 00202000400,00000000330034, 00202000500,00003003600035,
     * 00300000000,00005000150000, 00202000400,00000000360037,
     * 00202200000,00004000140040, 00202000600,00000000260000/
C  MISCELLANEOUS DECLARATIONS
      LOGICAL COMMNT
      DATA KNCOMM/1H*/
C
C  ONCE-ONLY CODE
C
      CALL DATE(9,TXDATE)
      CALL TOD(8,TXTIME)
      CALL KLOCK
      CPTIME = 0.
      WRITE(6,8) TXTIME,TXDATE
    8 FORMAT(1H1,30X,6HTIME   ,A6,A2,5X,6HDATE   ,A6,A3//)
      KKARD(4) = 1
C
C  READ CONTROL CARD AND INTERPRET
C
  100 IF(QCCRD) READ(5,101,END=200) KARD
  101 FORMAT(14A6)
      QCCRD = .TRUE.
      IF (KKARD(4).EQ.2) GO TO 112
      KTCD = KTCD+1
      QCCPRT = PRTCCD
C   * IN CC1 MEANS COMMENT
      COMMNT = KOM(KARD,1,1,KNCOMM,1).EQ.0
      IF (COMMNT) GO TO 106
      IF(.NOT.PRTCCD) GO TO 110
C  LIST CONTROL CARD IF SWITCH IS ON
  106 WRITE(6,FCC) PC,KTCD,KARD
      PC = PS
      IF (COMMNT) GO TO 100
C  SEARCH FOR C.C. NAME, AND INTERPRET CARD
  110 KSYN = 1
      LSYN = 72
  112 I = KKARD(4)
      KKARD(4) = 1
      IF(SYNTAX(KARD,KSYN,LSYN,XSYN,I,TSYN,CCARD,SSYN,NSSYN,KKARD))
     * GO TO 119
```

```
      IF(KTYPE.LE.0) WHY(1)=.TRUE.                                      TRAG0810
      IF(KTYPE.GT.0) WHY(KTYPE)=.TRUE.                                  TRAG0820
      IF(.NOT.OCCPRT) WRITE(6,FCC) PC,KTCD,KARD                         TRAG0830
      OCCPRT = .TRUE.                                                   TRAG0840
      IF(KKARD(13).GT.0) KSYN=KKARD(13)                                 TRAG0850
      WRITE(6,111) (SYNB,I=1,KSYN),SYND                                 TRAG0860
  111 FORMAT(17X,100A1)                                                 TRAG0870
      WRITE (6,113)                                                     TRAG0880
  113 FORMAT(44H ******  ERROR IN ABOVE CONTROL CARD  ******)            TRAG0890
      CALL FLUSH                                                        TRAG0900
      GO TO 100                                                         TRAG0910
  119 IF(KKARD(4).GE.3) GO TO 100                                       TRAG0920
      IF (KKARD(4).EQ.2) GCCRD=.FALSE.                                  TRAG0930
      GO TO (100,200,200,400,500,600,700,800,900                        TRAG0940
     * ),KTYPE                                                          TRAG0950
C                                                                       TRAG0960
C  END OF PROCESSING                                                    TRAG0970
C                                                                       TRAG0980
  200 STOP                                                              TRAG0990
C                                                                       TRAG1000
C  COMPUTE BASIC FRAGMENTATION INFO. TABLE                              TRAG1010
C                                                                       TRAG1020
  400 CALL LINK1(3)                                                     TRAG1030
      GO TO 9000                                                        TRAG1040
C                                                                       TRAG1050
C  DEFINE TYPES OF OUTPUT FUNCTIONS.  ALL MUST YIELD VALUES AT (R,PHI)  TRAG1060
C  DEPENDENT ONLY ON R, PHI, AND FRAG. INFO AT (R,PHI).                 TRAG1070
C                                                                       TRAG1080
  500 CALL LINK2                                                        TRAG1090
      GO TO 9000                                                        TRAG1100
C                                                                       TRAG1110
C  POSITION A FILE                                                      TRAG1120
C                                                                       TRAG1130
  600 CALL TAPE                                                         TRAG1140
      GO TO 9000                                                        TRAG1150
C                                                                       TRAG1160
C  READ, NORMALIZE, SMOOTH (AND LIST) ORDNANCE DATA                     TRAG1170
C                                                                       TRAG1180
  700 CALL LINK1(1)                                                     TRAG1190
      GO TO 9000                                                        TRAG1200
```

```
C                                                                          TRAG1210
C  READ AND PRINT CLOCK                                                    TRAG1220
C                                                                          TRAG1230
  800 CALL KLOCK(DUM)                                                      TRAG1240
      DUM1 = DUM-CPTIME                                                    TRAG1250
      IF (PC.EQ.PS) PC=PD                                                  TRAG1260
      WRITE (6,801) PC,DUM,DUM1                                            TRAG1270
  801 FORMAT(A1,33X,27H******  C.P. TIME   TOTAL =,F8.3,9H SEC  (UP,       TRAG1280
     * F7.3,13H SEC)  ******)                                              TRAG1290
      CPTIME = DUM                                                         TRAG1300
      PC = PD                                                              TRAG1310
      GO TO 100                                                            TRAG1320
C                                                                          TRAG1330
C  UNUSED STATEMENTS                                                       TRAG1340
C                                                                          TRAG1350
  900 CONTINUE                                                             TRAG1360
      GO TO 200                                                            TRAG1370
C                                                                          TRAG1380
C  AFTER EVERY COMMAND, TEST ABORT FLAG BEFORE READING NEXT COMMAND        TRAG1390
C                                                                          TRAG1400
 9000 IF(WHY(2)) GO TO 200                                                 TRAG1410
      GO TO 100                                                            TRAG1420
C                                                                          TRAG1430
      END                                                                  TRAG1440
```

```
      BLOCK DATA                                                                BLK 0010
      COMMON /CNTRL/ SCCRD,SCCPRT,KTCD,FCC(8),KARD(14),PC,PS,PD,PF,             BLK 0020
     * TXTIME(2),TXDATE(2),WHY(20),KKARD(800)                                   BLK 0030
      DATA SCCRD,KTCD,FCC,PC,PS,PD,PF,WHY/.TRUE., 0,                            BLK 0040
     * 26H(A1,8HCARD NO.,I5,5X,14A6),3*0,                                       BLK 0050
     * 1H ,1H ,1H0,1H1, 20*.FALSE./                                             BLK 0060
      DATA (KKARD (I),I=1, 124)/10*0,  76,9*0,                                  BLK 0070
     * 1, 797, 501, 501, 504,   1,   0, 2, 777, 505, 505, 504,   1,   0,BLK 0080
     * 2, 777,   0,   0,   0,   0,   0, 4, 761, 505, 505, 512,   1,   0,BLK 0090
     * 5, 737, 513, 513, 525,   1,   0, 6, 685, 526, 527, 537,   1,   0,BLK 0100
     * 7, 637, 538, 538, 551,   1,   0, 8, 589, 552, 552, 551,   1,   0,BLK 0110
     *     48*0/                                                                BLK 0120
      DATA (KKARD (I),I= 125, 276)/                                             BLK 0130
     *    81000, 593.0, 0,0,0,                        0.0,                      BLK 0140
     *    81000, 597.0, 0,0,0,                        0.0,                      BLK 0150
     *    81000, 601.0, 0,0,0,                        0.0,                      BLK 0160
     *        1, 605.0, 0,0,0,     .FALSE.            .0,                       BLK 0170
     *        2, 609.0, 0,0,0,     5                  .0,                       BLK 0180
     *        1, 613.0, 0,0,0,     .FALSE.            .0,                       BLK 0190
     *        1, 617.0, 0,0,0,     .FALSE.            .0,                       BLK 0200
     *        1, 621.0, 0,0,0,     .FALSE.            .0,                       BLK 0210
     *        1, 625.0, 0,0,0,     .FALSE.            .0,                       BLK 0220
     *    82002, 629.0, 0,0,0,                        0.0,                      BLK 0230
     *        2, 633.0, 0,0,0,     1                  .0,                       BLK 0240
     *    81000, 641.0, 0,0,0,                        0.0,                      BLK 0250
     *    81000, 645.0, 0,0,0,                        0.0,                      BLK 0260
     *    81000, 649.0, 0,0,0,                        0.0,                      BLK 0270
     *        1, 653.0, 0,0,0,     .FALSE.            .0,                       BLK 0280
     *        1, 657.0, 0,0,0,     .FALSE.            .0,                       BLK 0290
     *        3, 661.0, 0,0,0, 6H       ,6H            ,                        BLK 0300
     *        3, 665.0, 0,0,0, 6H       ,6H            ,                        BLK 0310
     *        1, 669.0, 0,0,0,     .FALSE.            .0/                       BLK 0320
      DATA (KKARD (I),I= 277, 428)/                                             BLK 0330
     *        1, 673.0, 0,0,0,     .FALSE.            .0,                       BLK 0340
     *        1, 677.0, 0,0,0,     .FALSE.            .0,                       BLK 0350
     *        2, 681.0, 0,0,0,     0                  .0,                       BLK 0360
     *    82002,   0.0, 0,0,0,                        0.0,                      BLK 0370
     *    81000, 689.0, 0,0,0,                        0.0,                      BLK 0380
     *    81000, 693.0, 0,0,0,                        0.0,                      BLK 0390
     *    81000, 697.0, 0,0,0,                        0.0,                      BLK 0400
```

```
     *         1, 701.0, 0,0,0,     .FALSE.      ,0,                   BLK 0410
     *         1, 705.0, 0,0,0,     .FALSE.      ,0,                   BLK 0420
     *         1, 709.0, 0,0,0,     .FALSE.      ,0,                   BLK 0430
     *         1, 713.0, 0,0,0,     .FALSE.      ,0,                   BLK 0440
     *         1, 717.0, 0,0,0,     .FALSE.      ,0,                   BLK 0450
     *         1, 721.0, 0,0,0,     .FALSE.      ,0,                   BLK 0460
     *         1, 725.0, 0,0,0,     .FALSE.      ,0,                   BLK 0470
     *         1, 729.0, 0,0,0,     .FALSE.      ,0,                   BLK 0480
     *         1, 733.0, 0,0,0,     .FALSE.      ,0,                   BLK 0490
     *         2, 741.0, 0,0,0,     36           ,0,                   BLK 0500
     *         2, 745.0, 0,0,0,     18           ,0,                   BLK 0510
     *         2, 749.0, 0,0,0,     0            ,0/                   BLK 0520
      DATA (KKARD (I),I= 429, 500)/                                    BLK 0530
     *         5, 753.0, 0,0,0,     0            ,0,                   BLK 0540
     *         2, 757.0, 0,0,0,     0            ,0,                   BLK 0550
     *      2003, 773.0, 0,0,0, 6H000    ,6H     ,                     BLK 0560
     *         5, 765.0, 0,0,0,     250,         ,0,                   BLK 0570
     *         2, 769.0, 0,0,0,     16           ,0,                   BLK 0580
     *     81000, 781.0, 0,0,0,                 0,0,                   BLK 0590
     *     81000, 785.0, 0,0,0,                 0,0,                   BLK 0600
     *         1, 789.0, 0,0,0,     .FALSE.      ,0,                   BLK 0610
     *         1, 793.0, 0,0,0,     .FALSE.      ,0/                   BLK 0620
      DATA (KKARD (I),I= 501, 588)/                                    BLK 0630
     *     493, 485, 477, 469, 461, 453, 445, 437, 429, 421, 413, 405, BLK 0640
     *     397, 389, 381, 373, 365, 357, 437, 349, 341, 333, 325, 317, BLK 0650
     *     309, 301, 293, 285, 277, 269, 261, 253, 245, 237, 229, 221, BLK 0660
     *     213, 205, 197, 189, 181, 173, 165, 437, 157, 373, 349, 149, BLK 0670
     *     141, 133, 125,                                              BLK 0680
     *      37*0/                                                      BLK 0690
      DATA (KKARD (I),I= 589, 740)/                                    BLK 0700
     *    6HTIM   ,6H      , -3,  0,   6HZZ7   ,6H      ,  3,  0,      BLK 0710
     *    6HYY7   ,6H      ,  3,  0,   6HXX7   ,6H      ,  3,  0,      BLK 0720
     *    6HSUM   ,6H      , -3,  0,   6HREAD  ,6H      , -4,  0,      BLK 0730
     *    6HSMOUTH,6H      , -6,  0,   6HNORM  ,6H      , -4,  0,      BLK 0740
     *    6HREZONE,6H      , -6,  0,   6HREDUCE,6H      , -6,  0,      BLK 0750
     *    6HFOR   ,6H      , -3,  0,   6HNSET  ,6H      , -4,  0,      BLK 0760
     *    6HORP   ,6H      , -3,  0,   6HZZ6   ,6H      ,  3,  0,      BLK 0770
     *    6HYY6   ,6H      ,  3,  0,   6HXX6   ,6H      ,  3,  0,      BLK 0780
     *    6HVE    ,6H      , -2,  0,   6HBK    ,6H      , -2,  0,      BLK 0790
     *    6HSR    ,6H      , -2,  0,   6HXS    ,6H      , -2,  0,      BLK 0800
```

```
     *     6HRE    ,6H      , -2,    0,    6HBI     ,6H      , -2,    0,          BLK 0810
     *     6HRC    ,6H      , -2,    0,    6HSK     ,6H      , -2,    0,          BLK 0820
     *     6HTAPE  ,6H      ,  4,    0,    6HZ75    ,6H      ,  3,    0,          BLK 0830
     *     6HYY5   ,6H      ,  3,    0,    6HXX5    ,6H      ,  3,    0,          BLK 0840
     *     6HLIM   ,6H      , -3,    0,    6HPA     ,6H      , -2,    0,          BLK 0850
     *     6HKW    ,6H      , -2,    0,    6HTA     ,6H      , -2,    0,          BLK 0860
     *     6HPL    ,6H      , -2,    0,    6HLIST   ,6H      , -4,    0,          BLK 0870
     *     6HEX    ,6H      , -2,    0,    6HRA     ,6H      , -2,    0,          BLK 0880
     *     6HSU    ,6H      , -2,    0,    6HOUT    ,6H      , -3,    0/          BLK 0890
      DATA (KKARD (I),I= 741, 800)/                                              BLK 0900
     *     6HMA    ,6H      , -2,    0,    6HNF     ,6H      , -2,    0,          BLK 0910
     *     6HMB    ,6H      , -2,    0,    6HZ      ,6H      , -1,    0,          BLK 0920
     *     6HOR    ,6H      , -2,    0,    6HBOOM   ,6H      ,  4,    0,          BLK 0930
     *     6HCR    ,6H      , -2,    0,    6HNR     ,6H      , -2,    0,          BLK 0940
     *     6HFL    ,6H      , -2,    0,    6HSTOP   ,6H      ,  4,    0,          BLK 0950
     *     6HYY1   ,6H      ,  3,    0,    6HXX1    ,6H      ,  3,    0,          BLK 0960
     *     6HDA    ,6H      , -2,    0,    6HCD     ,6H      , -2,    0,          BLK 0970
     *     6HPRINT ,6H      ,  5,    0/                                          BLK 0980
      END                                                                        BLK 0990
```

```
      SUBROUTINE FLUSH
C
C     IN THE EVENT OF AN ERROR, DATA CARDS ARE SKIPPED UNTIL THE NEXT
C     CONTROL CARD IS FOUND. (ALL CONTROL CARDS HAVE A '$' IN CC.1)
C     COMMENT CARDS ('*' IN CC.1) ARE LISTED
C
      COMMON /CNTRL/ QCCRD,QCCPRT,KTCD,FCC(8),KARD(14),PC,PS,PD,PF,
     * TXTIME(2),TXDATE(2),WHY(20),KKARD(600)
      LOGICAL QCCRD,QCCPRT,WHY
      INTEGER TXTIME,TXDATE,FCC,FC,PC,PD,PP
C
      LOGICAL Q
      DATA IPE,IRS/1H$,1H*/
C
      PC = PP
      Q = .FALSE.
    1 READ (5,2,END=200) KARD
    2 FORMAT(14A6)
      IF (KOM(KARD,1,1,IPE,1).EQ.0) GO TO 20
      IF (KOM(KARD,1,1,IRS,1).EQ.0) GO TO 10
      IF(Q) GO TO 1
      Q = .TRUE.
      WRITE(6,4) KARD
    4 FORMAT(1H0,18X,14A6)
      WRITE(6,5)
    5 FORMAT(35H0******  INPUT DATA SKIPPED  ******)
      GO TO 1
C  PRINT COMMENTS AS THEY APPEAR
   10 WRITE(6,4) KARD
      PC = PS
      GO TO 1
C  SET 'CONTROL CARD READY' FLAG WHEN CONTROL CARD IS FOUND
   20 PC = PP
      QCCRD = .FALSE.
      RETURN
C  CAUSE PROGRAM TERMINATION IF END OF DATA ENCOUNTERED.
  200 WHY(2) = .TRUE.
      RETURN
      END
```

```
      SUBROUTINE TAPE
C
C     GENERAL FILE POSITIONING ROUTINE
C
C     FOR FILE SEARCH COMMANDS, IT IS ASSUMED THAT THERE ARE SENTINAL
C     RECORDS, WHICH CONSIST OF THREE WORDS..
C          1)  6H*HEAD*
C          2)  1ST 6 CHARACTERS OF IDENTIFICATION (LEFT-JUST.)
C          3)  LAST 2 CHARACTERS OF IDENTIFICATION (BLANK IF ID IS
C              NOT OVER 6 CHARACTERS LONG)
C     THE IDENTIFICATION OF THE END-OF-FILE SENTINEL IS 6H999999,2H99,
C     WRITTEN JUST PRIOR TO AN END-OF-FILE.
C     THE 'SRCH' COMMAND SEARCHES FROM THE CURRENT FILE POSITION TO END
C     OF FILE.  THE 'XSRCH' COMMAND PROCEEDS AS 'SRCH', BUT REWINDS AND
C     SEARCHES THE WHOLE FILE IF AN END-OF-FILE IS READ.
C     IN BOTH CASES, AN ERROR IS PRINTED, AND WHY(6) IS SET IF THE
C     DESIRED SENTINAL RECORD IS NOT FOUND.
C     OPERATIONS ARE PERFORMED IN THE FOLLOWING ORDER IF MORE THAN ONE
C     OPERATION IS SPECIFIED..
C          1)  REWIND
C          2)  SKIP=N
C          3)  SR=ID OR XS=ID
C          4)  BKSP
C          5)  WEOF
C
      COMMON /CNTRL/ QCCRD,QCCPRT,KTCD,FCC(8),KARD(14),PC,PS,PD,PP,
     * TXTIME(2),TXDATE(2),WHY(20),KKARD(800)
      LOGICAL QCCRD,QCCPRT,WHY
      INTEGER TXTIME,TXDATE,FCC,PC,PS,PD,PP
C
      INTEGER ISRC(2),IXSRC(2)
      LOGICAL IFMODE,IFBIN,IFRWD,IFBKSP,IFWEOF,IFSRWD
      EQUIVALENCE (KKARD(297),IFSKIP),(KKARD(289),IFMODE),
     * (KKARD(281),IFBIN),(KKARD(273),IFRWD),(KKARD(263),KXSRC),
     * (KKARD(265),IXSRC),(KKARD(255),KSRC),(KKARD(257),ISRC),
     * (KKARD(249),IFBKSP),(KKARD(241),IFWEOF),(KKARD(303),KFILE),
     * (KKARD(305),IFILE)
C
      INTEGER IFSARG(2),IFLAR(2),IHEND(2)
      DATA IFSARG/2*1H /,IFHEAD,IHEND/6H*HEAD*,6H999999,2H99/
```

```
      DATA NWW,NCN/2,8/                                         TAPE0410
C                                                               TAPE0420
      IF(KFILE.LE.0) GO TO 640                                  TAPE0430
      IF(IFILE.LE.0) GO TO 640                                  TAPE0440
      IF(IFILE.EQ.5 .OR. IFILE.EQ.6) GO TO 640                  TAPE0450
      IF(IFMODE .AND.IFBIN) GO TO 640                           TAPE0460
      IF(KSRC*KXSRC.GT.0) GO TO 640                             TAPE0470
      WHY(6) = .FALSE.                                          TAPE0480
C  REWIND IF REQUESTED.                                         TAPE0490
      IF (IFRWD) REWIND IFILE                                   TAPE0500
      IF (IFSKIP.LE.0) GO TO 620                                TAPE0510
C  SKIP RECORDS IF REQUESTED.                                   TAPE0520
      IF (IFMODE) GO TO 610                                     TAPE0530
      DO 601 I=1,IFSKIP                                         TAPE0540
  601 READ (IFILE) IDUM                                         TAPE0550
      GO TO 620                                                 TAPE0560
  610 DO 611 I=1,IFSKIP                                         TAPE0570
  611 READ (IFILE,612) IDUM                                     TAPE0580
  612 FORMAT(A1)                                                TAPE0590
C  SEARCH FOR SENTINEL RECORD                                   TAPE0600
  620 IF(KSRC.LE.0 .AND. KXSRC.LE.0) GO TO 650                  TAPE0610
      IF(KSRC.GT.0) CALL MOVE(IFSARG,1,NCN,ISRC,1)              TAPE0620
      IF(KXSRC.GT.0) CALL MOVE(IFSARG,1,NCN,IXSRC,1)            TAPE0630
      IFSRWD = .FALSE.                                          TAPE0640
  621 IF(IFMODE) GO TO 622                                      TAPE0650
      READ (IFILE,END=630) IDUM,IFLAR                           TAPE0660
      GO TO 625                                                 TAPE0670
  622 READ (IFILE,624,END=630) IDUM,IFLAR                       TAPE0680
  624 FORMAT(3A6)                                               TAPE0690
  625 IF(IDUM.NE.IFHEAD) GO TO 621                              TAPE0700
      IF (KOMW(IFSARG,1,NCN,IFLAR).NE.0) GO TO 621              TAPE0710
  650 IF(IFBKSP) BACK SPACE IFILE                               TAPE0720
      IF(.NOT.IFWEOF) GO TO 100                                 TAPE0730
      IF(IFMODE) GO TO 651                                      TAPE0740
      WRITE(IFILE) IFHEAD,IHEND                                 TAPE0750
      GO TO 652                                                 TAPE0760
  651 WRITE(IFILE,624) IFHEAD,IHEND                             TAPE0770
  652 END FILE IFILE                                            TAPE0780
      GO TO 100                                                 TAPE0790
  630 IF(IFSRWD .OR. KXSRC.LE.0) GO TO 635                      TAPE0800
```

```
      IFSRWD = .TRUE.                                                   TAPE0810
      REWIND IFILE                                                      TAPE0820
      GO TO 621                                                         TAPE0830
  635 IF(.NOT.OCCPRT) WRITE(6,FCC) PC,KTCD,KARD                         TAPE0840
      OCCPRT = .TRUE.                                                   TAPE0850
      WRITE (6,636) IFSARG                                              TAPE0860
  636 FORMAT(19H0******  SENTINAL (,A6,A2,12H) NOT FOUND./)             TAPE0870
      GO TO 645                                                         TAPE0880
  640 IF(.NOT.OCCPRT) WRITE(6,FCC) PC,KTCD,KARD                         TAPE0890
      OCCPRT = .TRUE.                                                   TAPE0900
      WRITE(6,641)                                                      TAPE0910
  641 FORMAT(48H0******  INCONSISTENT OR INCORRECT PARAMETER(S)./)      TAPE0920
  645 WHY(6) = .TRUE.                                                   TAPE0930
      PC = PC                                                           TAPE0940
  100 RETURN                                                            TAPE0950
      END                                                               TAPE0960
```

```
      SUBROUTINE LINK1(L)                                               LINK0010
C                                                                       LINK0020
C     CALL LINK 1 ROUTINES                                              LINK0030
C                                                                       LINK0040
      IF (L-2) 10,20,30                                                 LINK0050
   10 CALL ORDNAN                                                       LINK0060
      RETURN                                                            LINK0070
   20 CALL EXIT                                                         LINK0080
      RETURN                                                            LINK0090
   30 CALL BOOM                                                         LINK0100
      RETURN                                                            LINK0110
      END                                                               LINK0120
```

```
      BLOCK DATA                                                        BLKB0010
C     ORDNANCE TABLES                                                   BLKB0020
      COMMON /ORDCOM/ MVORD,MMORD,IFORD,IVO,XKORD,                      BLKB0030
     * TTLORD,NVORD,NMORD,DTF,DTF2,                                     BLKB0040
     * VORD(37),AMORD(30),DORD(37,30),WORD(37,30)                       BLKB0050
      INTEGER TTLORD(12)                                                BLKB0060
      DATA NVORD,NMORD/37,30/                                           BLKB0070
      END                                                               BLKB0080
```

```
      SUBROUTINE ORDNAN
C
C  CONTROL CARD TABLES AND GENERAL PROGRAM CONTROL VARIABLES
      COMMON /CNTRL/ QCCRD,QCCPRT,KTCD,FCC(8),KARD(14),PC,PS,PD,PP,
     * TXTIME(2),TXDATE(2),WHY(20),KKARD(800)
      LOGICAL QCCRD,QCCPRT,WHY
      INTEGER TXTIME,TXDATE,FCC,PC,PS,PD,PP
C
      INTEGER AFFORD
      LOGICAL PRTDAT,QCBORD,REZONE,NRMORD,SMORD,LIST
      LOGICAL QWKPLT,SUMRY
      EQUIVALENCE (KKARD(489),PRTDAT),(KKARD(209),NSETS),
     * (KKARD(201),IFRORD),(KKARD(193),QCBORD),(KKARD(185),REZONE),
     * (KKARD(177),NRMORD),(KKARD(169),SMORD),(KKARD(161),INU),
     * (KKARD(377),LIST),(KKARD(441),AFFORD),(KKARD(353),QWKPLT),
     * (KKARD(153),SUMRY)
C
C  ORDNANCE TABLES
      COMMON /ORDCM/ NMORD,MMORD,IDORD,IVO,XKORD,
     * TTLORD,NVORD,NMORD,DTH,DTH2,
     * VORD(37),AMORD(30),PORD(37,30),WORD(37,30)
      INTEGER TTLORD(12)
C
      REAL COFORD(21)
      EXTERNAL ORDPLT
      REAL ZAREA(36),DTZ(36),WTZ(36),WAZ(36),WTM(30),DTM(30),DAM(30),
     * SDW(30),SDD(30),ORG(4,30),ORGXX(122)
      INTEGER CHOP(9)
      DATA CHOP/1HV,1HD,1HW,1HD,1HW,1HD,1HW,1HD,1HW/
      DATA PI/3.14159265/,MXFORD/10/
      DATA TXWGT,TXNO/6HWGT. (,5H) NO./
      EQUIVALENCE (ORGXX(1),ORG(1,1)),(ORGXX(121),VMN),
     * (ORGXX(122),VMX)
C
      WHY(7) = .FALSE.
      IF(INU.NE.5 .AND. (WHY(1).OR.WHY(6))) RETURN
      SUMRY=SUMRY.OR.LIST.OR.QWKPLT
      READ (5,160) TTLORD
  160 FORMAT(12A6)
      READ (INU,161) NMORD,NVORD,IDORD,IVO,XKORD
```

```
  161 FORMAT(3I5,1X,A1,3X,F10.0)
      IF(IVO.EQ.PS) IVO=PD
      WRITE(6,201) TTLORD,NMORD,NVORD,IDORD,IVO,XKORD
  201 FORMAT(1H0,18X,12A6//19X,7HNMORD =,I3,5X,7HNVORD =,I3,5X,
     * 6HI.D. =,I6,1H-,A1,5X,12HAVG GR/IN2 =,F10.2)
      IF (NMORD.LE.0 .OR. NVORD.LE.0 .OR. NVORD.GT.MVORD .OR.
     * XKORD.LE.0.) STOP
      IF(NSETS.LE.0) NSETS=1
      IF(IFRORD.EQ.1) CALL ORDIN1(NMORD,NVORD,NSETS,WORD,DORD,VORD,
     * QCBORD,PRTDAT,KTCD,INU,MVORD)
      IF(IFRORD.EQ.2) CALL ORDIN2(NMORD,NVORD,NSETS,WORD,DORD,VORD,
     * QCBORD,PRTDAT,KTCD,INU,MVORD)
      PC = PD
      IF (PRTDAT) PC=PF
      IF(.NOT.REZONE) GO TO 210
C   IF REDUCTION OF ZONES IS REQUESTED
      IDUM = NVORD-1
      DO 230 I=1,NMORD
      DORD(1,I) = DORD(1,I)*2.
      DORD(NVORD,I) = DORD(NVORD,I)*2.
      DO 231 J=1,IDUM
      DUM = DORD(J,I)+DORD(J+1,I)
      IF (DUM.LE.0.) GO TO 232
      WORD(J,I) = (WORD(J,I)*DORD(J,I)+WORD(J+1,I)*DORD(J+1,I))/DUM
      GO TO 231
  232 WORD(J,I) = 0.
  231 DORD(J,I) = DUM/2.
  230 CONTINUE
      NVORD = IDUM
  210 DTH = PI/FLOAT(NVORD)
      DTH2=DTH/2.
      DPHP = 180./FLOAT(NVORD)
C   GENERATE POLAR ZONE AREA (ON UNIT SPHERE)
      DUM = DTH2
      PDUM = 4.*PI*SIN(DUM)
      DO 211 I=1,NVORD
      ZAREA(I) = SIN(DUM)*PDUM
  211 DUM = DUM+DTH
      IF(.NOT.NRMORD) GO TO 220
C   NORMALIZE
```

```
      DO 215 I=1,NVORD
      DO 215 J=1,NWORD
  215 DORD(I,J) = DORD(I,J)/ZAREA(I)
C  CLEAR STD. DEVIATION CELLS
  220 SDV = 0.
      DO 249 I=1,NWORD
      SDW(I) = 0.
  249 SDD(I) = 0.
      IF(.NOT.SMORD) GO TO 270
C  SMOOTH VORD, DORD, AND WORD
C  1) TRY TO *FILL IN* WORD(*,*)=0 ENTRIES
      DO 221 I=1,NWORD
      DUM = 0.
      J = 1
  228 IF(WORD(J,I).GT.0.) GO TO 222
      IF(J.GT.1) DUM=WORD(J-1,I)
      IF(J.GE.NVORD) GO TO 241
      DO 223 K=J,NVORD
      IF(WORD(K,I).GT.0.) GO TO 224
  223 CONTINUE
C  A) TRAILING ZEROES -- SET TO LAST NON-ZERO VALUE (IF ANY)
  241 DO 225 K=J,NVORD
  225 WORD(K,I)=DUM
      GO TO 221
  224 K1 = K-1
      IF(J.GT.1) GO TO 226
C  B) LEADING ZEROES -- SET TO 1ST NON-ZERO VALUE
      DO 227 L=1,K1
  227 WORD(L,I) = WORD(K,I)
      J = K
      GO TO 222
C  C) INTERMEDIATE ZEROES -- FILL IN VIA LINEAR INTERPOLATION
  226 DUM = WORD(J-1,I)
      DK = (WORD(K,I)-DUM)/FLOAT(K-J+1)
      DO 242 L=J,K1
      DUM = DUM+DK
  242 WORD(L,I) = DUM
      J = K
  222 J = J+1
  229 IF(J.LE.NVORD) GO TO 228
```

```
  221 CONTINUE                                                    ORDN1210
C   2) SMOOTH ALL CURVES USING FOURIER SERIES                     ORDN1220
C      NOTE,, MASS AND NUMBER NOT CONSERVED EXACTLY BY THESE FITS ORDN1230
       IF (AFFORD.LE.0) GO TO 270                                 ORDN1240
       AFFORD = MIN0(AFFORD,MXFORD)                               ORDN1250
       CALL FOUR(VORD,COFORD,NVORD,AFFORD)                        ORDN1260
       CALL FORGET(WTZ,COFORD,NVORD,AFFORD)                       ORDN1270
       DO 246 J=1,NVORD                                           ORDN1280
       SDV = SDV+(WTZ(J)-VORD(J))**2                              ORDN1290
  246  VORD(J) = WTZ(J)                                           ORDN1300
       SDV = SQRT(SDV/FLOAT(NVORD))                               ORDN1310
       DO 243 I=1,NMORD                                           ORDN1320
       CALL   FOUR(WORD(1,I),COFORD,NVORD,AFFORD)                 ORDN1330
       CALL FORGET(WTZ,COFORD,NVORD,AFFORD)                       ORDN1340
       DO 244 J=1,NVORD                                           ORDN1350
       SDW(I) = SDW(I)+(WTZ(J)-WORD(J,I))**2                      ORDN1360
  244  WORD(J,I) = WTZ(J)                                         ORDN1370
       SDW(I) = SQRT(SDW(I)/FLOAT(NVORD))                         ORDN1380
       CALL   FOUR(DORD(1,I),COFORD,NVORD,AFFORD)                 ORDN1390
       CALL FORGET(WTZ,COFORD,NVORD,AFFORD)                       ORDN1400
       DO 245 J=1,NVORD                                           ORDN1410
       SDD(I) = SDD(I)+(WTZ(J)-DORD(J,I))**2                      ORDN1420
  245  DORD(J,I) = WTZ(J)                                         ORDN1430
       SDD(I) = SQRT(SDD(I)/FLOAT(NVORD))                         ORDN1440
  243  CONTINUE                                                   ORDN1450
C  DETERMINE VARIOUS AVERAGES AND TOTALS                          ORDN1460
  270  VA = 0.                                                    ORDN1470
       WT = 0.                                                    ORDN1480
       DT = 0.                                                    ORDN1490
       VMN = +1.E+38                                              ORDN1500
       VMX = -VMN                                                 ORDN1510
       DO 273 I=1,NMORD                                           ORDN1520
       ORG(1,I) = VMN                                             ORDN1530
       ORG(2,I) = VMX                                             ORDN1540
       ORG(3,I) = VMN                                             ORDN1550
  273  ORG(4,I) = VMX                                             ORDN1560
       DO 271 J=1,NVORD                                           ORDN1570
       DUM = AMAX1(VORD(J),0.)                                    ORDN1580
       VMN = AMIN1(VMN,DUM)                                       ORDN1590
       VMX = AMAX1(VMX,DUM)                                       ORDN1600
```

Reproduced from best available copy.

```
      WTZ(J) = 0.                                                       ORDN1610
      DTZ(J) = 0.                                                       ORDN1620
      DO 272 I=1,NMORD                                                  ORDN1630
      UNIT = AMAX1(DORD(J,I),0.)                                        ORDN1640
      COUNT = UNIT*ZAREA(J)                                             ORDN1650
      WEIGHT = AMAX1(WORD(J,I),0.)                                      ORDN1660
      WTZ(J) = WTZ(J)+COUNT*WEIGHT                                      ORDN1670
      DTZ(J) = DTZ(J)+COUNT                                             ORDN1680
      ORG(1,I) = AMIN1(ORG(1,I),WEIGHT)                                 ORDN1690
      ORG(2,I) = AMAX1(ORG(2,I),WEIGHT)                                 ORDN1700
      ORG(3,I) = AMIN1(ORG(3,I),UNIT)                                   ORDN1710
  272 ORG(4,I) = AMAX1(ORG(4,I),UNIT)                                   ORDN1720
      WAZ(J) = WTZ(J)/DTZ(J)                                            ORDN1730
      VA = VA+VORD(J)*WTZ(J)                                            ORDN1740
      WT = WT+WTZ(J)                                                    ORDN1750
  271 DT = DT+DTZ(J)                                                    ORDN1760
      VA = VA/WT                                                        ORDN1770
      WA = WT/DT                                                        ORDN1780
      DO 274 I=1,NMORD                                                  ORDN1790
      DAM(I) = 0.                                                       ORDN1800
      WTM(I) = 0.                                                       ORDN1810
      DTM(I) = 0.                                                       ORDN1820
      DO 275 J=1,NVORD                                                  ORDN1830
      WEIGHT = AMAX1(WORD(J,I),0.)                                      ORDN1840
      UNIT = AMAX1(DORD(J,I),0.)                                        ORDN1850
      COUNT = UNIT*ZAREA(J)                                             ORDN1860
      WTM(I) = WTM(I)+WEIGHT*COUNT                                      ORDN1870
  275 DTM(I) = DTM(I)+COUNT                                             ORDN1880
      DAM(I) = DTM(I)/(4.*PI)                                           ORDN1890
  274 AMORD(I) = WTM(I)/DTM(I)                                          ORDN1900
C  LIST FINAL DATA IN COLUMNS IF REQUESTED                              ORDN1910
  280 IF(.NOT.LIST) GO TO 250                                           ORDN1920
      M2 = MIN0(5,NVORD)                                                ORDN1930
      WRITE (6,281) PC,(TXWGT,I,TXND,I=1,M2)                            ORDN1940
  281 FORMAT(A1,63X,15HMASS CATEGORIES/                                 ORDN1950
     * 12HD POLAR ZONE,12X,8HVELOCITY,5(7X,A6,I2,A5))                   ORDN1960
      PHP1 = 0.                                                         ORDN1970
      PC = PD                                                           ORDN1980
      DO 282 J=1,NVORD                                                  ORDN1990
      PHP2 = PHP1+DPHP                                                  ORDN2000
```

```
      WRITE(6,283) PC,PHP1,PHP2,WORD(J),(WORD(J,I),CORD(J,I),I=1,M2)    ORDN2010
  283 FORMAT(A1,F5.1,1H-,F5.1,13X,11F10.1)                               ORDN2020
      PHP1 = PHP2                                                        ORDN2030
  282 PC = PS                                                            ORDN2040
  284 M1 = M2+1                                                          ORDN2050
      IF(M2.GE.NMORD) GO TO 289                                          ORDN2060
      M2 = MIN0(NMORD,M1+5)                                              ORDN2070
      PC = PP                                                            ORDN2080
      WRITE(6,285) PC,(TX,GT,I,TXNC,I=M1,M2)                             ORDN2090
  285 FORMAT(A1,63X,15H ASS CATEGORIES/                                  ORDN2100
     * 12H  POLAR ZONE,6(7X,A6,I2,A5))                                   ORDN2110
      PHP1 =0.                                                           ORDN2120
      PC = PP                                                            ORDN2130
      DO 286 J=1,NVORD                                                   ORDN2140
      PHP2 = PHP1+DPHP                                                   ORDN2150
      WRITE(6,287) PC,PHP1,PHP2,(NORD(J,I),DORD(J,I),I=M1,M2)            ORDN2160
  287 FORMAT(A1,F5.1,1H-,F5.1,12F10.1)                                   ORDN2170
      PHP1 = PHP2                                                        ORDN2180
  286 PC = PS                                                            ORDN2190
      GO TO 284                                                          ORDN2200
  289 PC = PP                                                            ORDN2210
C  LIST SUMMARY OF DATA IF REQUESTED                                     ORDN2220
  250 IF (.NOT.SUMRY) GO TO 260                                          ORDN2230
      WRITE(6,257)                                                       ORDN2240
  257 FORMAT(1H1,57X,15HSUMMARY OF DATA//                                ORDN2250
     * 12H  POLAR ZONE,5X,10HTOTAL MASS,7X,8HNO. FRAG,7X,8HAVG MASS/)    ORDN2260
      PHP1 = 0.                                                          ORDN2270
      PC = PP                                                            ORDN2280
      DO 256 J=1,NVORD                                                   ORDN2290
      PHP2 = PHP1+DPHP                                                   ORDN2300
      WRITE(6,251) PHP1,PHP2,WTZ(J),CTZ(J),WAZ(J)                        ORDN2310
  251 FORMAT(1X,F5.1,1H-,F5.1,3F15.1)                                    ORDN2320
  256 PHP1 = PHP2                                                        ORDN2330
      WRITE(6,252) WT,CT,WA                                              ORDN2340
  252 FORMAT(12H0  ALL ZONES,3F15.1//)                                   ORDN2350
      WRITE(6,253) VMN,VA,VMX,SDV                                        ORDN2360
  253 FORMAT(1H1,53X,23HSUMMARY OF DATA (CONT.)//                        ORDN2370
     * 34X,3HLOW,8X,7HAVERAGE,11X,4HHIGH,8X,7HSTD DEV,10X,5HTOTAL//      ORDN2380
     * 14X,8HVELOCITY,4F15.1)                                            ORDN2390
      DO 254 I=1,NMORD                                                   ORDN2400
```

```
254 WRITE(6,255) I,ORG(3,I),FAM(I),ORG(4,I),SDF(I),DTM(I),          ORDN2410
   *               ORG(1,I),AMORD(I),ORG(2,I),SDW(I),WTM(I)          ORDN2420
255 FORMAT(11HOMASS CAT, ,I2,9H NO. FRAG,5F15.1/18X,4HMASS,5F15.1)   ORDN2430
260 IF(.NOT.SWKPLT) GO TO 100                                        ORDN2440
    NPLOT = NMORD/4+1                                                ORDN2450
    AZ1 = DPHP/2.                                                    ORDN2460
    AZ2 = 180.-AZ1                                                   ORDN2470
    CALL ORDPLT(1,F+36,PC,ONE)                                       ORDN2480
    PC = PP                                                          ORDN2490
    M1 = 1                                                           ORDN2500
    KC = 1                                                           ORDN2510
    DO 261 I=1,NPLOT                                                 ORDN2520
    M2 = MIN0(NMORD,M1+3)                                            ORDN2530
    IF (I.EQ.1) M2=MIN0(M2,3)                                        ORDN2540
262 WRITE(6,263) PC,M1,M2                                            ORDN2550
263 FORMAT(A1,10X,47HPLOT OF FINAL DENSITY AND WEIGHT TABLES, MASSES, ORDN2560
   * I3,5H THRU,I3/)                                                 ORDN2570
    NF = 2*(M2-M1+1)                                                 ORDN2580
    IF(I.EQ.1) NF=NF+1                                               ORDN2590
    CALL NOTARY(ORDPLT,NF,NMORD,AZ1,AZ2,121,0.,6.,CHOP(KC))          ORDN2600
    KC = 2                                                           ORDN2610
    M1 = M2+1                                                        ORDN2620
    IF(I.NE.NPLOT) CALL ORDPLT(-1,,0,DUM)                            ORDN2630
261 CONTINUE                                                         ORDN2640
100 PC = PP                                                          ORDN2650
    IF(PRTDAT .OR. SURVEY) PC=PP                                     ORDN2660
    RETURN                                                           ORDN2670
    END                                                              ORDN2680
```

```
      SUBROUTINE ORDIN1(WORD,MVORD,NSETS,WORD,DORD,VORD,QCBORD,PRTDAT,  ORDI0010
     * KTCD,INU,MVORD)                                                   ORDI0020
C                                                                        ORDI0030
C     INPUT ROUTINE FOR FORMAT=1 DISTANCE DATA                           ORDI0040
C***  NSETS.GT.1 AND QCBORD .T. NOT SUPPORTED                            ORDI0050
C                                                                        ORDI0060
      REAL WORD(MVORD,1),DORD(MVORD,1),VORD(MVORD)                       ORDI0070
      LOGICAL QCBORD,PRTDAT                                              ORDI0080
C                                                                        ORDI0090
      DO 10 J=1,MVORD                                                    ORDI0100
      READ(INU,2)  (WORD(J,I),DORD(J,I),I=1,NWORD)                       ORDI0110
    2 FORMAT(4(2F5.1,5X))                                                ORDI0120
      IF (PRTDAT) WRITE(6,4) (WORD(J,I),DORD(J,I),I=1,NWORD)             ORDI0130
    4 FORMAT(10X,10HWORD,DORD ,4(2F10.3,4X))                             ORDI0140
      READ(INU,3) VORD(J)                                                ORDI0150
    3 FORMAT(50X,F10.1)                                                  ORDI0160
      IF(PRTDAT) WRITE(6,5) VORD(J)                                      ORDI0170
    5 FORMAT(10X,4HVORD,6X,F12.1)                                        ORDI0180
   10 CONTINUE                                                           ORDI0190
      RETURN                                                             ORDI0200
      END                                                                ORDI0210
```

```
      SUBROUTINE ORDINP(NWORD,NVORD,NSETS,WORD,DORD,VORD,QCBORD,PRTDAT,
     * KTCP,INU,MVORD)
C
C     INPUT ROUTINE FOR FORMAT=2 ORDNANCE DATA
C
C***  NOTE THAT WZ AND WZD SHOULD BE DIMENSIONED FOR COL.LEN=MVORD.
      REAL WORD(MVORD,1),DORD(MVORD,1),VORD(MVORD),WZ(37,2),WZD(37),
     * WT(4),DT(4)
      LOGICAL QCBORD,QNWORD,PRTDAT
C
      NUM = NWORD
      IF (QCBORD) NUM=(NUM+1)/2
      QNWORD = QCBORD .AND. NWORD.NE.NWORD/2*2
C  CLEAR ORD. ARRAYS
      DO 2 J=1,NVORD
      WZD(J)=0.
      VORD(J) = 0.
      DO 3 I=1,2
    3 WZ(J,I) = 0.
      DO 4 I=1,NUM
      WORD(J,I) = 0.
    4 DORD(J,I) = 0.
    2 CONTINUE
C  READ ORD, MASS AND NUMBER TABLES
      DO 11 N=1,NSETS
      IF(PRTDAT) WRITE(6,82) N
   82 FORMAT(10X,23HWORD,DORD FOR COMPONENT,I2)
      I1 = 1
   16 I2 = I1+1
      IF(.NOT.QCBORD) I2=I2+2
      I2 = MIN0(NUM,I2)
      DO 12 J=1,NVORD
      READ (INU,13) (WT(K),DT(K),K=1,4)
   13 FORMAT(4(2F5.0,5X))
      IF (PRTDAT) WRITE(6,81) (WT(K),DT(K),K=1,4)
   81 FORMAT(20X,4(2F10.3,4X))
      K = 0
      DO 14 I=I1,I2
      K = K+1
      IF(.NOT.QCBORD) GO TO 15
```

```
      K = K+1                                                           ORDI0410
      WORD(J,I) = WORD(J,I)+WT(K-1)*T(K-1)                              ORDI0420
      DORD(J,I) = DORD(J,I)+DT(K-1)                                     ORDI0430
      WZ(J,N) = WZ(J,N)+WT(K-1)*DT(K-1)                                 ORDI0440
      IF(I.EQ.NUM .AND. NMORD) GO TO 14                                 ORDI0450
   15 WORD(J,I) = WORD(J,I)+WT(K)*T(K)                                  ORDI0460
      DORD(J,I) = DORD(J,I)+DT(K)                                       ORDI0470
      WZ(J,N) = WZ(J,N)+WT(K)*DT(K)                                     ORDI0480
   14 CONTINUE                                                          ORDI0490
   12 CONTINUE                                                          ORDI0500
      I1 = I2+1                                                         ORDI0510
      IF(I2.LT.NUM) GO TO 16                                            ORDI0520
   11 CONTINUE                                                          ORDI0530
      NMORD = NUM                                                       ORDI0540
C  SET AVG MASS ARRAYS (DIVIDE TOTAL MASS(J,I) BY TOT. NO. (J,I)  )    ORDI0550
      DO 20 J=1,NVORD                                                   ORDI0560
      DO 20 I=1,NMORD                                                   ORDI0570
      IF(DORD(J,I).LE.0.) GO TO 22                                      ORDI0580
      WORD(J,I) = WORD(J,I)/DORD(J,I)                                   ORDI0590
      GO TO 20                                                          ORDI0600
   22 WORD(J,I) = 0.                                                    ORDI0610
   20 CONTINUE                                                          ORDI0620
C  READ AVG VEL,S(J,N), WEIGHT BY TOTAL MASSES                          ORDI0630
      DO 23 N=1,NSETS                                                   ORDI0640
      IF (PRTDAT) WRITE(6,83) N                                         ORDI0650
   83 FORMAT(10X,18HVORD FOR COMPONENT,I2)                              ORDI0660
      DO 21 J=1,NVORD                                                   ORDI0670
      READ(INU,24) VT                                                   ORDI0680
   24 FORMAT(50X,F10.0)                                                 ORDI0690
      IF (PRTDAT) WRITE(6,84) VT                                        ORDI0700
   84 FORMAT(20X,F10.1)                                                 ORDI0710
      WZD(J) = WZD(J)+WZ(J,N)                                           ORDI0720
   21 VORD(J) = VT*WZ(J,N)+VORD(J)                                      ORDI0730
   23 CONTINUE                                                          ORDI0740
C  STORE OVERALL AVG V'S -- CONSERVE MOMENTUM                           ORDI0750
      DO 30 J=1,NVORD                                                   ORDI0760
      IF(WZD(J).LE.0.) GO TO 31                                         ORDI0770
      VORD(J) = VORD(J)/WZD(J)                                          ORDI0780
      GO TO 30                                                          ORDI0790
   31 VORD(J) = 0.                                                      ORDI0800
   30 CONTINUE                                                          ORDI0810
      RETURN                                                            ORDI0820
      END                                                               ORDI0830
```

```
      SUBROUTINE ORDPLT(X,N,V)                                          ORDP0010
      REAL V(1),DM(30),DA(30),WM(30),WA(30)                             ORDP0020
C  ORDNANCE TABLES                                                      ORDP0030
      COMMON /ORDCM/ NVORD,NMORD,IORD,IVO,XKORD,                        ORDP0040
     * TTLORD,NVORD,NMORD,DTH,DTH2,                                     ORDP0050
     * VORD(37),AMORD(30),DORD(37,30),WORD(37,30)                       ORDP0060
      INTEGER TTLORD(12)                                                ORDP0070
C                                                                       ORDP0080
C     SUPPLY FUNCTION VALUES (PLOTTING ORDNANCE DATA)                   ORDP0090
C     X = 1 TO NVORD NORMALLY.  IF X.GE.1.E+38, INITIALIZE EVERYTHING.  ORDP0100
C     IF X.LE.0., INITIALIZE NEW PLOT.                                  ORDP0110
C                                                                       ORDP0120
      IF(X.GE.1.E+38) GO TO 10                                          ORDP0130
      IF(X.LT.0.) GO TO 20                                              ORDP0140
      I = I+1                                                           ORDP0150
      K = 1                                                             ORDP0160
      IF(NM1.GT.1) GO TO 2                                              ORDP0170
      V(1) = AMAX1(0.,VORD(I))*VM+VA                                    ORDP0180
      K = 2                                                             ORDP0190
    2 DO 1 J=NM1,NM2                                                    ORDP0200
      V(K) = AMAX1(0.,DORD(I,J))*DM(J)+DA(J)                            ORDP0210
      V(K+1) = AMAX1(0.,WORD(I,J))*WM(J)+WA(J)                          ORDP0220
    1 K = K+2                                                           ORDP0230
      RETURN                                                            ORDP0240
   20 NM1 = NM2+1                                                       ORDP0250
      NM2 = MINO(NM1+3,NVORD)                                           ORDP0260
      RETURN                                                            ORDP0270
   10 NM1 = 1                                                           ORDP0280
      NM2 = MINO(3,NMORD)                                               ORDP0290
      YMN = 0.                                                          ORDP0300
      YMX = 8.0                                                         ORDP0310
      DY = 2.04                                                         ORDP0320
      DP = 1.82                                                         ORDP0330
      K = 1                                                             ORDP0340
      DO 11 I=1,NMORD                                                   ORDP0350
      WA(I) = V(K)                                                      ORDP0360
      WM(I) = V(K+1)                                                    ORDP0370
      DA(I) = V(K+2)                                                    ORDP0380
      DM(I) = V(K+3)                                                    ORDP0390
   11 K = K+4                                                           ORDP0400
```

```
      K = 4*NMORD                                   ORDP0410
      VA = V(K+1)                                   ORDP0420
      VM = V(K+2)                                   ORDP0430
      VM = DP/(VM-VA)                               ORDP0440
      VA = YMN-VA*VM                                ORDP0450
      Y = YMN+DY                                    ORDP0460
      DO 14 J=1,NMORD                               ORDP0470
      DM(J) = DP/(DM(J)-DA(J))                      ORDP0480
      DA(J) = Y-DA(J)*DM(J)                         ORDP0490
      WM(J) = DP/(WM(J)-WA(J))                      ORDP0500
      WA(J) = Y-WA(J)*WM(J)                         ORDP0510
      Y = Y+DY                                      ORDP0520
      IF(Y.GE.YMX) Y=YMN                            ORDP0530
   14 CONTINUE                                      ORDP0540
      I = 0                                         ORDP0550
      RETURN                                        ORDP0560
      END                                           ORDP0570
```

```
      SUBROUTINE FOUR(F,C,N,M)
C
C     FIND FOURIER COEFFICIENTS THRU ORDER M, STORE IN ARRAY C
C     C(1) IS BIAS,  C(2*K) IS SIN-K COEF,  C(2*K+1) IS COS-K COEF.
C     FUNCTION F IS DEFINED AT MIDPOINTS OF N EQUI-SPACED INTERVALS.
C
      REAL F(1),C(1)
      DATA PI/3.1415926/
C
      N21 = M+M+1
      DO 1 I=1,N21
    1 C(I) = 0.
      DX = PI/FLOAT(N)
      SX = SIN(DX)
      CX = COS(DX)
      SDX = 2.*SX*CX
      CDX = CX*CX-SX*SX
      DO 2 I=1,N
      C(1) = C(1)+F(I)
      IF(M.LE.0) GO TO 5
      SNX = SX
      CNX = CX
      DO 3 K=1,M
      C(2*K  ) = C(2*K  )+F(I)*SNX
      C(2*K+1) = C(2*K+1)+F(I)*CNX
      DUM = SNX*CX+CNX*SX
      CNX = CNX*CX-SNX*SX
      SNX = DUM
    3 CONTINUE
    5 DUM = SX*CDX+CX*SDX
      CX = CX*CDX-SX*SDX
      SX = DUM
    2 CONTINUE
      C(1) = C(1)/PI*DX
      IF(M.LE.0) RETURN
      DO 4 I=2,N21
    4 C(I) = 2.*C(I)/PI*DX
      RETURN
      END
```

```
      SUBROUTINE FORGET(F,C,N,M)                                        FORG0010
C                                                                       FORG0020
C     GIVEN FOURIER COEFFICIENTS THRU ORDER M, COMPUTE FUNCTION VALUES  FORG0030
C     AT MIDPOINTS OF N EQUI-SPACED INTERVALS.  COEFFICIENTS ARE        FORG0040
C     IN C. STORE FUNCTION VALUES IN F.                                 FORG0050
C                                                                       FORG0060
      REAL F(1),C(1)                                                    FORG0070
      DATA PI/3.14159265/                                               FORG0080
C                                                                       FORG0090
      DUM = PI/FLOAT(N)                                                 FORG0100
      SX = SIN(DUM)                                                     FORG0110
      CX = COS(DUM)                                                     FORG0120
      SDX = 2.*SX*CX                                                    FORG0130
      CDX = CX*CX-SX*SX                                                 FORG0140
      DO 2 I=1,N                                                        FORG0150
      F(I) = C(1)                                                       FORG0160
      IF(M.LE.0) GO TO 5                                                FORG0170
      SNX = SX                                                          FORG0180
      CNX = CX                                                          FORG0190
      DO 3 K=1,M                                                        FORG0200
      F(I) = F(I)+C(2*K)*SNX+C(2*K+1)*CNX                               FORG0210
      DUM = SNX*CX+CNX*SX                                               FORG0220
      CNX = CNX*CX-SNX*SX                                               FORG0230
      SNX = DUM                                                         FORG0240
    3 CONTINUE                                                          FORG0250
    5 DUM = SX*CDX+CX*SDX                                               FORG0260
      CX = CX*CDX-SX*SDX                                                FORG0270
      SX = DUM                                                          FORG0280
    2 CONTINUE                                                          FORG0290
      RETURN                                                            FORG0300
      END                                                               FORG0310
```

```
      SUBROUTINE BOOM
C
C     GIVEN TRAJECTORY TABLE AND ORDNANCE DATA, COMPUTES FRAGMENT FIELD.
C     ORDNANCE DATA IS IN /ORDCM/.
C     TRAJECTORY TABLE IS IN /TRAJCM/
C     FRAGMENT FIELD IS WRITTEN OUT ON FILE 'IFRTAP'
C     FIELD IS COMPUTED ON A HORIZONTAL HALF-CIRCLE OF RADIUS
C     'NRNG*DRNG' AND AT AN ALTITUDE OF 'Z'
C     LINEAR AND CIRCULAR BARRIERS MAY BE PLACED IN THE FIELD.
C
C  CONTROL CARD TABLES AND GENERAL PROGRAM CONTROL VARIABLES
      COMMON /CNTRL/ QCCRD,QCCPRT,KTCD,FCC(8),KARD(14),PC,PS,PD,PP,
     * TXTIME(2),TXDATE(2),WHY(20),KKARD(800)
      LOGICAL QCCRD,QCCPRT,WHY
      INTEGER TXTIME,TXDATE,FCC,PC,PS,PD,PP
C
      LOGICAL PRTDAT
      EQUIVALENCE (KKARD(469),PRTDAT),(KKARD(457),DRNG),
     * (KKARD(449),IPOSOP),(KKARD(433),Z),
     * (KKARD(425),NBAR),(KKARD(417),NALF),(KKARD(409),NRN),
     * (KKARD(465),NRNG)
C
C  ORDNANCE TABLES
      COMMON /ORDCM/ NVORD,NMORD,IPORD,IVO,XKORD,
     * TTLORD,NVORD,NMORD,DTH,DTH2,
     * VORD(37),AMORD(30),DORD(37,30),WORD(37,30)
      INTEGER TTLORD(12)
      EQUIVALENCE (NM,NMORD)
C
      DIMENSION PACK(20,180),KPK1(60),KPK2(60),ALFHI(30),
     * TEMP(4,30,40),KPTA(30),KPTD(30),IDIP(2),IPOSOP(2)
      DIMENSION IBTYP(1),BCPH(1),BVPH(1),BHAL(1),TAHAL(1)
      DIMENSION CL(3),CH(3)
      LOGICAL QBAR,QBARP-
      INTEGER TTLFRG(12)
      DATA PI,RADEG/3.14159265,0.0174532925/
      DATA IHDR,IDIP/6H*HEAD*,6H00000 ,2H  /
C     IFRTAP   FRAGMENT FIELD OUTPUT TAPE
C     NWN      NUMBER OF WORDS IN A TAPE I.D.
      DATA IFRTAP/4/,NWN/2/
```

```
C      MXPH      MAX NO AZIMUTH RAYS ALLOWED                              BOOM0410
C      MXRNG     MAX NO RANGES ALLOWED                                    BOOM0420
C      MXRAY     MAX ALLOWED NO TRAJ. CALCS PER AZ RAY                    BOOM0430
       DATA  MXPH,MXRNG,MXRAY/36,20,40/                                   BOOM0440
C      EPM1, EPM2    FITTING TOLERANCES FOR 'CFIT'                        BOOM0450
       DATA EPM1,EPM2/0.,0.5/                                             BOOM0460
C      EPS       HEIGHT TOLERANCE FOR CONVERGENCE, 1-STEP INTEGRATION     BOOM0470
C      MKOUNT    MAX NO ITERATIONS ALLOWED, 1-STEP INTEGRATION            BOOM0480
       DATA EPS/0.1/,MKOUNT/10/                                           BOOM0490
C      G         ACCELERATION OF GRAVITY, FT/SEC**2                       BOOM0500
C      G2        HALF OF G                                                BOOM0510
C      RETAO     DRAG COEF * AIR DENS * 7000 / (2*144)                    BOOM0520
       DATA G,G2,RETAO/32.17,16.085,2.3269/                               BOOM0530
C      CL,CH     VALUE LIMITS FOR OUTPUT VALUES.. VEL.. ANGLE, DENSITY    BOOM0540
       DATA CL/0.,-1.5708,0./,CH/1.E+38,+1.5708,1.E+38/                   BOOM0550
C                                                                         BOOM0560
       WHY(4) = WHY(7) .OR. WHY(3)                                        BOOM0570
       IF (NPH.LE.0 .OR. NPH.GT.MXPH .OR. NALF.LE.0 .OR. NALF.GT.MXRAY    BOOM0580
     * .OR. NRNG.LE.0 .OR. NRNG.GT.MXRAY .OR. NRNG.GT.MXRNG .OR.          BOOM0590
     * DRNG.LE.0.) GO TO 910                                              BOOM0600
       KITEMP = MMORD*4                                                   BOOM0610
       IVA = PD                                                           BOOM0620
       IVF = PD                                                           BOOM0630
       CALL MOVE(IVA,1,1,IPOSOP,1)                                        BOOM0640
       CALL MOVE(IVF,1,1,IPOSOP,2)                                        BOOM0650
       IF (IVA.EQ.PS) IVA=PD                                              BOOM0660
       IF (IVF.EQ.PS) IVF=PD                                              BOOM0670
       DO 212 I=1,NWN                                                     BOOM0680
  212  IPOSOP(I) = IDIP(I)                                                BOOM0690
       I = IFYO(IPOSOP,1,2,0,IDORD)                                       BOOM0700
       CALL MOVE(IPOSOP,3,1,IVO,1)                                        BOOM0710
       CALL MOVE(IPOSOP,4,1,IVA,1)                                        BOOM0720
       CALL MOVE(IPOSOP,5,1,IVF,1)                                        BOOM0730
       READ(5,210) TTLFRG                                                 BOOM0740
  210  FORMAT(12A6)                                                       BOOM0750
       WRITE (6,211) TTLFRG                                               BOOM0760
  211  FORMAT(1H0,18X,12A6)                                               BOOM0770
       WRITE (6,213) NPH,NALF,NRNG,DRNG,IPOSOP                            BOOM0780
  213  FORMAT(1H0,18X,5HNPH =,I3,5X,6HNALF =,I3,5X,6HNRNG =,I3,5X,        BOOM0790
     * 6HDRNG =,F6.1,5X,11HTAPE KEY = ,A6,A2)                             BOOM0800
```

```
C  READ BARRIER INFORMATION, IF INDICATED                              ROOM0810
      IF (NBAR.LE.0) GO TO 202                                          ROOM0820
      IF (NBAR.GT.1) GO TO 905                                          ROOM0830
      WRITE (6,224)                                                     ROOM0840
  224 FORMAT(1H0,18X,19HBARRIER INFORMATION)                            ROOM0850
      DO 229 I=1,NBAR                                                   ROOM0860
      READ(5,220) IBTYP(I),IDUM,BCPH(I),BWPH(I),BHAL(I)                 ROOM0870
  220 FORMAT(2I5,6F10.0)                                                ROOM0880
      IF(PRTDAT) WRITE(6,221) IBTYP(I),IDUM,BCPH(I),BWPH(I),BHAL(I)     ROOM0890
  221 FORMAT(19X,2I5,3E15.5)                                            ROOM0900
      BCPH(I) = BCPH(I)*RADEG                                           ROOM0910
      BWPH(I) = BWPH(I)*RADEG                                           ROOM0920
      BHAL(I) = BHAL(I)*RADE                                            ROOM0930
      TBHAL(I) = TAN(BHAL(I))                                           ROOM0940
  229 CONTINUE                                                          ROOM0950
C  RETURN IF ERROR FLAGS ARE SET.                                       ROOM0960
  202 IF(WHY(1).OR.WHY(4).OR.WHY(6)) GO TO 100                          ROOM0970
C  BEGIN FRAGMENT FIELD OUTPUT                                          ROOM0980
      WRITE (IFRTAP) IRDR,IPOSDR                                        ROOM0990
      WRITE (IFRTAP) TTLOFR,TTLFRS,NM,XKCRD,NRNG,DRNG,NPH               ROOM1000
      WRITE (IFRTAP) (XKCRD(I),I=1,NM)                                  ROOM1010
      IMX = 6*NM                                                        ROOM1020
      IMY = 2*NM                                                        ROOM1030
      DAL = PI/(2.*FLOAT(NALF))                                         ROOM1040
      DPH = PI/FLOAT(NPH)                                               ROOM1050
      DPH2 = DPH/2.                                                     ROOM1060
      DAL2 = DAL/2.                                                     ROOM1070
      PHI = DPH2                                                        ROOM1080
C                                                                       ROOM1090
C  COMPUTE FRAGMENT FIELD FOR EVERY AZIMUTH ANGLE (NPH OF THEM)         ROOM1100
C                                                                       ROOM1110
      DO 401 IPH=1,NPH                                                  ROOM1120
      DO 404 IM=1,NM                                                    ROOM1130
  404 KPTA(IM) = 0                                                      ROOM1140
      DO 403 IM=1,IMY                                                   ROOM1150
  403 KPK1(IM) = 0                                                      ROOM1160
      SPHI = SIN(PHI)                                                   ROOM1170
      CPHI = COS(PHI)                                                   ROOM1180
C  TEST FOR BARRIER(S) ALONG CURRENT AZIMUTH                            ROOM1190
      ALFBAR = 0.                                                       ROOM1200
```

```
      IF(NBAR.LE.0) GO TO 249                                           BOOM1210
      DO 248 I=1,NBAR                                                   BOOM1220
      DUM = ABS(PHI-BOPH(I))                                            BOOM1230
      IF(DUM.GT.BWPH(I)) GO TO 248                                      BOOM1240
      IF(IBTYP(I).NE.1) GO TO 240                                       BOOM1250
      ALFBAR = AMAX1(ALFBAR,BHAL(I))                                    BOOM1260
      GO TO 248                                                         BOOM1270
  240 ALFBAR = AMAX1(ALFBAR,ATAN(TBHAL(I)*COS(DUM)))                    BOOM1280
  248 CONTINUE                                                          BOOM1290
  249 QBARPH = ALFBAR.GT.0.                                             BOOM1300
C                                                                       BOOM1310
C  COMPUTE ASCENDING FRAGS AND DESC LOW-REG FRAGS, STEPPING RANGE       BOOM1320
C                                                                       BOOM1330
      DO 395 IM=1,NM                                                    BOOM1340
      ALF = 0.                                                          BOOM1350
      RNG = DRNG                                                        BOOM1360
      DO 393 IR=1,NRNG                                                  BOOM1370
      TEST2L = 1.E+35                                                   BOOM1380
      R2 = RNG*RNG                                                      BOOM1390
      U = 0.                                                            BOOM1400
      T = 0.                                                            BOOM1410
      H = Z                                                             BOOM1420
      DUM = SQRT(H*H+R2)                                                BOOM1430
      SAP = H/DUM                                                       BOOM1440
      CAP = RNG/DUM                                                     BOOM1450
C  1-STEP INTEGRATION METHOD -- ITERATE FOR SATISFACTORY INITIAL ALPHA  BOOM1460
      DO 315 ICOUNT=1,MCOUNT                                            BOOM1470
      TH = ACOS(CAP*CPHI)                                               BOOM1480
      CALL INTSET(NVORD,DTH2,DTH,TH,ITH,PTH)                            BOOM1490
      VORDI = AMAX1(0.,FINFT(VORD,ITH,PTH))                             BOOM1500
      WORDI = AMAX1(0.,FINFT(WORD(1,IM),ITH,PTH))                       BOOM1510
      BETA = BETA0/(YKORD*YKORD*VORDI)**0.333333                        BOOM1520
      XB = -G2*T*T*SAP*(1.+U/3.)/(1.+U)                                 BOOM1530
      YB = -G2*T*T*CAP*(U*(1.+U*0.5)-ALOG(1.+U))/(U*U)                  BOOM1540
      XBO = RNG/CAP+YB*SAP/CAP-XB                                       BOOM1550
      DUM = BETA*XBO                                                    BOOM1560
C***  ABORT ITERATION IF EXP(DUM) WOULD BE VERY LARGE OR SMALL          BOOM1570
      IF (ABS(DUM).GT.12.) GO TO 392                                    BOOM1580
      U = EXP(DUM)-1.                                                   BOOM1590
      T=U/(BETA*VORDI)                                                  BOOM1600
```

Reproduced from best available copy.

```
      XB = -G2*T*T*SAP*(1.+U/3.)/(1.+U)                                 BOOM1610
      YB = -G2*T*T*CAP*(U*(1.+U*0.5)-ALOG(1.+U))/(U*U)                  BOOM1620
      H = (XBO+XB)*SAP+YB*CAP                                           BOOM1630
      DUM1 = Z-H                                                        BOOM1640
      DUM = SQRT(DUM1*DUM1+R2)                                          BOOM1650
      SDA = DUM1/DUM                                                    BOOM1660
      CDA = RNG/DUM                                                     BOOM1670
      DUM = SAP*CDA+CAP*SDA                                             BOOM1680
      CAP = CAP*CDA-SAP*SDA                                             BOOM1690
      SAP = DUM                                                         BOOM1700
      TEST2=ABS(DUM1)                                                   BOOM1710
C  DIVERGENCE INDICATES THAT RNG.GT.MAX RANGE FOR GIVEN INIT. CONDS.    BOOM1720
      IF(TEST2.GT.TEST2L) GO TO 390                                     BOOM1730
C  CONVERGENCE IF TERM. HEIGHT CLOSE ENOUGH TO Z.                       BOOM1740
      IF (TEST2.LT.EPS) GO TO 320                                       BOOM1750
      TEST2L=TEST2                                                      BOOM1760
  315 CONTINUE                                                          BOOM1770
      GO TO 392                                                         BOOM1780
  320 XDO = VORDI/(1.+U)                                                BOOM1790
      XDB = -G*T*SAP*(1.+U*(1.+U/3.))/(1.+U)**2                         BOOM1800
      YDB = -G*T*CAP*(1.+U*0.5)/(1.+U)                                  BOOM1810
      XD = (XDO+XDB)*CAP-YDB*SAP                                        BOOM1820
      YD = (XDO+XDB)*SAP+YDB*CAP                                        BOOM1830
      VELT = SQRT(XD*XD+YD*YD)                                          BOOM1840
      DUM1 = XDO+XDB                                                    BOOM1850
      DUM = SQRT(YDB*YDB+DUM1*DUM1)                                     BOOM1860
      SDA = YDB/DUM                                                     BOOM1870
      CDA = DUM1/DUM                                                    BOOM1880
      SAF = SAP*CDA+CAP*SDA                                             BOOM1890
      CAF = CAP*CDA-SAP*SDA                                             BOOM1900
      DUM = 2.*SAP*CAP                                                  BOOM1910
      DRDA = (2.*RNG-XBO*CAP)*((CAP*CAP-SAP*SAP)/DUM-CAF/SAF)           BOOM1920
     *       -XBO*CAP/DUM                                               BOOM1930
      DORDI = AMAX1(0.,FINFT(DORD(*,1,1),ITH,PTH))                      BOOM1940
      ALF = ATAN2(SAP,CAP)                                              BOOM1950
      QBAR = QBARPH .AND. ALF.LE.ALFBAR                                 BOOM1960
      I = KPTA(IM)+1                                                    BOOM1970
      KPTA(IM) = I                                                      BOOM1980
      TEMP(1,IM,I) = RNG                                                BOOM1990
      TEMP(2,IM,I) = VELT                                               BOOM2000
```

```
      TEMP(3,IM,I) = ATAN2(SAF,CAF)                                      BOOM2010
      TEMP(4,IM,I) = 0.                                                  BOOM2020
C***  NOTE THAT UNIFORM-SOURCE FORM OF DR/D ALPHA IS USED                BOOM2030
      IF (.NOT.GRAR) TEMP(4,IM,I)=DORDI*CAP/ABS(RNG*DRDA*SAF)            BOOM2040
  390 RNG = RNG+DRNG                                                     BOOM2050
  392 ALFHI(IM) = ALF                                                    BOOM2060
  395 CONTINUE                                                           BOOM2070
C                                                                        BOOM2080
C  COMPUTE HIGH-REC DESCENDING FRAGS, STEPPING ELEVATION ANGLE           BOOM2090
C                                                                        BOOM2100
      ALFLO = PI                                                         BOOM2110
      DO 405 IM=1,NM                                                     BOOM2120
      KPTD(IM) = KPTA(IM)                                                BOOM2130
      ALFLO = AMIN1(ALFLO,ALFHI(IM))                                     BOOM2140
  405 CONTINUE                                                           BOOM2150
      KAL = (ALFLO+DAL2)/DAL+1.                                          BOOM2160
      ALF = FLOAT(KAL)*DAL-DAL2                                          BOOM2170
      DO 402 IAL=KAL,NALF                                                BOOM2180
      SALF = SIN(ALF)                                                    BOOM2190
      CALF = COS(ALF)                                                    BOOM2200
      GBAR = GRARPH .AND. ALF.LE.ALFRAR                                  BOOM2210
C  THETA IS POLAR ANGLE ON SPHERE (NOSE=0, TAIL=180.)                    BOOM2220
      CTH = CALF*CPHI                                                    BOOM2230
      TH = ACOS(CTH)                                                     BOOM2240
      STH = SQRT(1.-CTH*CTH)                                             BOOM2250
      DTHDAL = SALF*CPHI/STH                                             BOOM2260
C  PSI IS LONGITUDINAL ANGLE ON SPHERE, OBSERVED LOOKING FROM TAIL TO    BOOM2270
C  NOSE,  (UP=0,  AT AZIM=90, PSI=90-ELEV)                               BOOM2280
      SPSI = CALF*SPHI/STH                                               BOOM2290
      CPSI = SALF/STH                                                    BOOM2300
      CALL INTSET(NVORD,DTH2,DTH,TH,ITH,PTH)                             BOOM2310
      VORDI = FINFT(VORD,ITH,PTH)                                        BOOM2320
      VORDI = AMAX1(0.,VORDI)                                            BOOM2330
      DVORDI = FINFD(VORD,ITH,PTH,DTH)                                   BOOM2340
C  COMPUTE FIELD FOR EACH MASS CATEGORY                                  BOOM2350
      DO 410 IM=1,NMORP                                                  BOOM2360
      IF (ALF.LE.ALFHI(IM)) GO TO 410                                    BOOM2370
      WORDI = FINFT(WORD(1,IM),ITH,PTH)                                  BOOM2380
      WORDI = AMAX1(0.,WORDI)                                            BOOM2390
      DORDI = FINFT(DORD(1,IM),ITH,PTH)                                  BOOM2400
```

```
      DORDI = AMAX1(0.,DORDI)                                           BOOM2410
      CALL NSTEP(XKORD,WORDI,VORDI,-7,ALF,RNGT,VELT,SALFT,CALFT)        BOOM2420
      ALFT = ATAN2(SALFT,CALFT)                                         BOOM2430
      I = KPTD(IM)+1                                                    BOOM2440
      IF (I.GT.MXRAY) GO TO 900                                         BOOM2450
      KPTD(IM) = I                                                      BOOM2460
      TEMP(1,IM,I) = RNGT                                               BOOM2470
      TEMP(2,IM,I) = VELT                                               BOOM2480
      TEMP(3,IM,I) = ALFT                                               BOOM2490
      TEMP(4,IM,I) = 0.                                                 BOOM2500
C***  NOTE THAT DR/DA TAKEN AS DELTA R/DELTA ALPHA                      BOOM2510
      IF(.NOT.QBAR) TEMP(4,IM,I) = DORDI*CALF*DAL/ABS(RNGT*SALFT)       BOOM2520
  410 CONTINUE                                                          BOOM2530
  402 ALF = ALF+DAL                                                     BOOM2540
C                                                                       BOOM2550
C  COMPLETE PROCESSING OF TERMINAL CHARACTERISTICS                      BOOM2560
C                                                                       BOOM2570
      IMM = 0                                                           BOOM2580
      JM = 0                                                            BOOM2590
      DO 420 IM=1,NMORD                                                 BOOM2600
C  COMPLETE FRAGMENT DENSITY COMPUTATION FOR DESCENDING LEG OF TRAJ.    BOOM2610
      I1 = KPTA(IM)+1                                                   BOOM2620
      I2 = KPTD(IM)                                                     BOOM2630
      IF (I1.GT.I2) GO TO 465                                           BOOM2640
      DUM1 = TEMP(1,IM,1)                                               BOOM2650
      DUM = DAL*DUM1                                                    BOOM2660
      DO 461 I=I1,I2                                                    BOOM2670
      IF (I.EQ.I2) GO TO 462                                            BOOM2680
      DUM2 = TEMP(1,IM,I+1)                                             BOOM2690
      DUM = ABS(DUM2-DUM1)                                              BOOM2700
      DUM1 = DUM2                                                       BOOM2710
  462 TEMP(4,IM,I) = TEMP(4,IM,I)/DUM                                   BOOM2720
  461 CONTINUE                                                          BOOM2730
  465 IE = 0                                                            BOOM2740
  421 IE = IE+1                                                         BOOM2750
      GO TO (422,423,420),IE                                            BOOM2760
C  FOR ASC. AND LOW-REG. DESC. FRAGMENTS, INSERT INFO INTO OUTPUT TABLESBOOM2770
  422 IN = KPTA(IM)                                                     BOOM2780
      IF (IN.LE.0) GO TO 490                                            BOOM2790
      JM = JM+1                                                         BOOM2800
```

```
      KPK1(JM) = 1                                                      BOOM2810
      KPK2(JM) = IN                                                     BOOM2820
      DO 424 I=1,3                                                      BOOM2830
      IMM = IMM+1                                                       BOOM2840
      DO 424 J=1,IN                                                     BOOM2850
  424 PACK(J,IMM) = TEMP(I+1,IM,J)                                      BOOM2860
      GO TO 421                                                         BOOM2870
C  FOR DESCENDING HIGH-REG. FRAGMENTS, INTERPOLATE FOR OUTPUT INFO.     BOOM2880
  423 IF (KPTD(IM).LE.IN) GO TO 429                                     BOOM2890
      IS = IN+1                                                         BOOM2900
      II = KPTD(IM)                                                     BOOM2910
C  FIRST, REORDER TRAJ'S TO ORDER OF ASC. RANGE (FOR THE MOST PART)     BOOM2920
      J = II                                                            BOOM2930
      IDUM = IS+(II-IS-1)/2                                             BOOM2940
      DO 426 I=IS,IDUM                                                  BOOM2950
      DO 425 K=1,4                                                      BOOM2960
      DUM = TEMP(K,IM,I)                                                BOOM2970
      TEMP(K,IM,I) = TEMP(K,IM,J)                                       BOOM2980
  425 TEMP(K,IM,J) = DUM                                                BOOM2990
  426 J = J-1                                                           BOOM3000
C  ELIMINATE ANY LOW-REG. TRAJ'S.. I.E.. MAKE RANGE INCR. MONOTONICALLY BOOM3010
      J = II-1                                                          BOOM3020
      DO 427 I=IS,J                                                     BOOM3030
      IF (TEMP(1,IM,I+1).LE.TEMP(1,IM,I)) GO TO 428                     BOOM3040
  427 CONTINUE                                                          BOOM3050
      I = II                                                            BOOM3060
  428 II = I                                                            BOOM3070
      IN = II-IN                                                        BOOM3080
      IDUM = MAX0(1,MIN0(NRNG,TEMP(1,IM,IS)/DRNG+0.999999))             BOOM3090
      JDUM = MAX0(IDUM,MIN0(NRNG,TEMP(1,IM,II)/DRNG))                   BOOM3100
      JM = JM+1                                                         BOOM3110
      KPK1(JM) = IDUM                                                   BOOM3120
      KPK2(JM) = JDUM                                                   BOOM3130
      DO 430 I=1,3                                                      BOOM3140
      IMM = IMM+1                                                       BOOM3150
      RNG = DRNG*FLOAT(IDUM)                                            BOOM3160
      DO 431 J=IDUM,JDUM                                                BOOM3170
      DUM = FLINT(TEMP(1,IM,IS),KITEMP,TEMP(I+1,IM,IS),KITEMP,IN,RNG)   BOOM3180
      DUM = AMAX1(CL(I),AMIN1(CH(I),DUM))                               BOOM3190
      PACK(J,IMM) = DUM                                                 BOOM3200
```

```
      RNG = RNG+DRNG                                                    BOOM3210
  431 CONTINUE                                                          BOOM3220
  430 CONTINUE                                                          BOOM3230
      GO TO 421                                                         BOOM3240
C  IF NO DATA FOR A TRAJ. TYPE, FILL IN ZEROES                          BOOM3250
  490 DO 491 I=1,3                                                      BOOM3260
      IMM = IMM+1                                                       BOOM3270
      DO 492 J=1,NRNG                                                   BOOM3280
  492 PACK(J,IMM) = 0.                                                  BOOM3290
  491 CONTINUE                                                          BOOM3300
      JM = JM+1                                                         BOOM3310
      GO TO 421                                                         BOOM3320
  420 CONTINUE                                                          BOOM3330
C  OUTPUT SMOOTHED TERM. CHAR. FOR THIS AZ. RAY                         BOOM3340
      WRITE (IFRTAP) (KPK1(J),KPK2(J), J=1,IMY),                        BOOM3350
     * ((PACK(I,J),I=1,NRNG),J=1,IMX)                                   BOOM3360
  401 PHI = PHI+DPH                                                     BOOM3370
      GO TO 100                                                         BOOM3380
C                                                                       BOOM3390
C  ERRORS                                                               BOOM3400
C                                                                       BOOM3410
  900 IF (.NOT. OCCPRT) WRITE(6,FCC) PC,KTCD,KARD                       BOOM3420
      OCCPRT = .TRUE.                                                   BOOM3430
      WRITE (6,901) KPTA(IM),I,IPH,IM                                   BOOM3440
  901 FORMAT(22H0******  RAY OVERFLOW,,I4,10H R-POINTS,,I3,             BOOM3450
     * 20H TOTAL POINTS ON RAY,I3,10H, MASS CAT,I3)                     BOOM3460
      GO TO 990                                                         BOOM3470
  905 IF (.NOT.OCCPRT) WRITE(6,FCC) PC,KTCD,KARD                        BOOM3480
      OCCPRT = .TRUE.                                                   BOOM3490
      WRITE (6,906)                                                     BOOM3500
  906 FORMAT(27H0******  TOO MANY BARRIERS.)                            BOOM3510
      GO TO 990                                                         BOOM3520
  910 IF (.NOT.OCCPRT) WRITE(6,FCC) PC,KTCD,KARD                        BOOM3530
      OCCPRT = .TRUE.                                                   BOOM3540
      WRITE (6,911)                                                     BOOM3550
  911 FORMAT(21H0******  BAD VALUE(S))                                  BOOM3560
  990 CALL FLUSH                                                        BOOM3570
      WHY(4) = .TRUE.                                                   BOOM3580
  100 PC = PD                                                           BOOM3590
      RETURN                                                            BOOM3600
      END                                                               BOOM3610
```

```
      SUBROUTINE NSTEP(XK,XGP,VLO,HOR,FLVO,X,VL,SLV,CLV)              NSTE0010
C                                                                       NSTE0020
      DATA PI/3.141592/                                                 NSTE0030
      DATA NDT/10/,BETA0/2.3260/                                        NSTE0040
      DATA G/32.17/                                                     NSTE0050
C                                                                       NSTE0060
      BT = BETA0/((XK*XK*XGP)**0.333333)                                NSTE0070
      VST=SQRT(G/BT)                                                    NSTE0080
      TST=1./(BT*VST)                                                   NSTE0090
      VL = VLO                                                          NSTE0100
      X=0.                                                              NSTE0110
      Y=HOR                                                             NSTE0120
      T=0.                                                              NSTE0130
      DT=TST/FLOAT(NDT)                                                 NSTE0140
      SLV = SIN(FLVO)                                                   NSTE0150
      CLV = COS(FLVO)                                                   NSTE0160
    2 VL1=VL                                                            NSTE0170
      TH=BT*VL*DT                                                       NSTE0180
      TH1=1.+TH                                                         NSTE0190
      ALT=ALOG(TH1)                                                     NSTE0200
      UB=VL/TH1-G*SLV*DT*(TH1+TH*TH/3.)/TH1**2                          NSTE0210
      VB=-G*CLV*DT*(1.+TH/2.)/TH1                                       NSTE0220
      DXB1=VL*DT                                                        NSTE0230
      IF(BT.GT.0.) DXB1=ALT/BT                                          NSTE0240
      DXB=DXB1-G*SLV*DT*DT*(3.+TH)/(4.*TH1)                             NSTE0250
      FTHY=1.-TH/3.                                                     NSTE0260
      IF(TH.GT.1.E-04) FTHY=(TH*(2.+TH)-2.*ALT)/(2.*TH*TH)              NSTE0270
      DYB=-G*CLV*DT*DT*FTHY/2.                                          NSTE0280
      DX=DXB*CLV-DYB*SLV                                                NSTE0290
      DY=DXB*SLV+DYB*CLV                                                NSTE0300
      VL=SQRT(UB*UB+VB*VB)                                              NSTE0310
      UB = UB/VL                                                        NSTE0320
      VB = VB/VL                                                        NSTE0330
      DUM = SLV*UB+CLV*VB                                               NSTE0340
      CLV = CLV*UB-SLV*VB                                               NSTE0350
      SLV = DUM                                                         NSTE0360
      X=X+DX                                                            NSTE0370
      Y=Y+DY                                                            NSTE0380
      T=T+DT                                                            NSTE0390
      IF (Y.GT.0. .OR. DY.GT.0.) GO TO 2                                NSTE0400
```

```
      VL=VL-Y*(VL-VL1)/DY                                               NSTE0410
      DUM = -Y*ATAN2(VB,UB)/DY                                          NSTE0420
      VB = SIN(DUM)                                                     NSTE0430
      UB = COS(DUM)                                                     NSTE0440
      DUM = SLV*UB+CLV*VB                                               NSTE0450
      CLV = CLV*UB-SLV*VB                                               NSTE0460
      SLV = DUM                                                         NSTE0470
      X=X-Y*DX/DY                                                       NSTE0480
      T=T-Y*DT/DY                                                       NSTE0490
      RETURN                                                            NSTE0500
      END                                                               NSTE0510
```

```
      SUBROUTINE INTSET(N,XL,DX,Y,IX,PX)                                INTS0010
C                                                                       INTS0020
C  FIND MAIN ENTRY NO. 'IX' AND EX-FRACT. 'PX', GIVEN TABLE OF 'N'      INTS0030
C  ENTRIES AT 'DX' INTERVAL, STARTING AT 'XL'.  SEARCH VALUE IS 'Y'     INTS0040
C                                                                       INTS0050
      IX = (Y-XL)/DX+1.0                                                INTS0060
      IX = MAX0(2,MIN0(IX,N-2))                                         INTS0070
      PX = (Y-XL)/DX-FLOAT(IX-1)                                        INTS0080
      RETURN                                                            INTS0090
      END                                                               INTS0100
```

```
      FUNCTION FINFT(Y,I,P)                                              FINF0010
      REAL Y(1)                                                          FINF0020
C                                                                        FINF0030
C     4-POINT (-1,0,+1,+2) LAGRANGE INTERPOLATION                        FINF0040
C                                                                        FINF0050
      P1 = P*P-1                                                         FINF0060
      P2 = P-2.                                                          FINF0070
      FINFT = -P*(P-1.)*P2*Y(I-1)/6.+P1*P2*Y(I)/2.-P*(P+1.)*P2*Y(I+1)/2.FINF0080
     * + P*P1*Y(I+2)/6.                                                  FINF0090
      RETURN                                                             FINF0100
      END                                                                FINF0110
```

```
      FUNCTION FINFD(Y,I,P,D)                                            FINF0010
      REAL Y(1)                                                          FINF0020
C                                                                        FINF0030
C     4-PT (-1,0,+1,+2) LAGRANGE DIFFERENTIATION                         FINF0040
C                                                                        FINF0050
      P1 = 3.*P*P                                                        FINF0060
      FINFD = ( (-P1+6.*P-2.)*Y(I-1)/6.+(P1-4.*P-1.)*Y(I)/2.             FINF0070
     * -(P1-2.*P-2.)*Y(I+1)/2.+(P1-1.)*Y(I+2)/6.)/D                      FINF0080
      RETURN                                                             FINF0090
      END                                                                FINF0100
```

```
      FUNCTION FLINT(XK,KX,YK,KY,N,X)
      REAL XK(1),YK(1)
C
C     LINEAR INTERPOLATION ROUTINE.
C     THERE ARE N VALUE PAIRS (XK(IX),YK(IY)), WHERE IX=1,1+KX, ETC.,
C     AND IY=1,1+KY, ETC.  THE ARGUMENT IS X.
C     RETURN 0 IF X IS OUT OF RANGE.  RETURN YK(1) IF N=1.
C
      JX = 1+(N-1)*KX
      IF (X.LT.XK(1).OR.X.GT.XK(JX)) GO TO 12
      IF (N.LT.2) GO TO 10
      JX = 1
      DO 1 J=2,N
      JX = JX+KX
      IF (X.LE.XK(JX)) GO TO 2
    1 CONTINUE
      J = N
    2 LX = JX
      JX = JX-KX
      JY = 1+(J-2)*KY
      LY = JY+KY
      FLINT = YK(JY)+(X-XK(JX))*(YK(LY)-YK(JY))/(XK(LX)-XK(JX))
      RETURN
C   ONLY ONE VALUE IN TABLE.  RETURN IT.
   10 FLINT = YK(1)
      RETURN
C   X IS OUT OF RANGE.  RETURN 0.
   12 FLINT = 0.
      RETURN
      END
```

```
      SUBROUTINE LINK2
C
C     GIVEN FRAGMENT FIELD DATA GENERATED BY [illegible], COMPUTE PROBABILITY
C     FUNCTIONS AND OTHER FUNCTIONS OF THE FIELD PARAMETERS.
C     FRAGMENT FIELD INPUT IS READ FROM FILE 'FFDTAP'
C     FUNCTION VALUE TABLES ARE WRITTEN ON FILE 'FFLTAP', FOR USE BY
C     A PLOT GENERATION PROGRAM.
C     FUNCTION VALUE CONTOUR PLOTS [illegible] MAY BE GENERATED
C     ON THE PRINTER, FOR QUICK-LOOK RESULTS.
C
C  CONTROL CARD TABLES AND GENERAL PROGRAM CONTROL VARIABLES
      COMMON /CNTRL/ QCCRD,QCCPRT,KTCD,FCC(8),KARD(14),PC,PS,PD,PF,
     * TXTIME(2),TXDATE(2),WHY(20),KKARD(800)
      LOGICAL QCCRD,QCCPRT,WHY
      INTEGER TXTIME,TXDATE,FCC,PC,PS,PD,PF
C
      LOGICAL PRTDAT,LIST,QWKPLT,EXEC
      EQUIVALENCE (KKARD(489),PRTDAT),(KKARD(377),LIST),
     * (KKARD(353),QWKPLT),(KKARD(385),EXEC)
C
      IF(EXEC) GO TO 110
C
C  READ FUNCTION PARAMETERS, TABLES, ETC.
C
      CALL OUTSET
      GO TO 100
C
C  COMPUTE AND OUTPUT FUNCTIONS
C
  110 CALL COMPUT
      IF (WHY(5)) GO TO 100
      CALL OUTPUT
C
  100 PC = PS
      IF(LIST.OR.PRTDAT.OR.WHY(5)) PC=PD
      IF (QWKPLT) PC=PF
      RETURN
      END
```

```
      BLOCK DATA
C
C  FUNCTION TABLES, PARAMETERS AND VALUE STORAGE
      COMMON /PLOTCM/ NRNG,DRNG,NPH,FPH,DPH2,IVC,IVA,IVF,IPOSOP(2),
     * NSETS,IPORD,
     * TTLORD(12),TTLFRG(12),NPLOTS,NX,NY,XMIN,YMIN,DX,DY,NZ,CH(55),
     * RECT(81,41),ITF(20),ISF(20),ITYF(20),IVPF(20),NF(20),F(14,20),
     * ITTLF(12,20),KTF(20),FZMIN(20),FZMAX(20),XTB(14,10),YTB(14,10),
     * C(7,10),NTB(10),KTB(10),OUT(20,36,8),ITTLP(12,8),KTPP(8),OLP(8),
     * OHP(8),ISP(8),ITP(8),PZMIN(8),PZMAX(8),ITYP(8),ITYS(8),IVPF(8)
C
      DATA KTF,FZMIN,FZMAX/20*0,20*-1.E+30,20*1.E+30/
      DATA NZ,CH/10,1H ,1H ,1H ,1H-,1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,
     * 1H8,1H9,1H+,40*0/
C
      END
```

```
      SUBROUTINE OUTSET                                                 OUTS0010
C                                                                       OUTS0020
C  CONTROL CARD TABLES AND GENERAL PROGRAM CONTROL VARIABLES            OUTS0030
      COMMON /CNTRL/ GCORD,GCCPRT,KTCD,FCC(8),KARD(14),PC,PS,PD,PP,     OUTS0040
     * TXTIME(2),TXDATE(2),WHY(20),KKARD(600)                           OUTS0050
      LOGICAL GCORD,GCCPRT,WHY                                          OUTS0060
      INTEGER TXTIME,TXDATE,FCC,PC,PS,PD,PP                             OUTS0070
C                                                                       OUTS0080
      LOGICAL PRTDAT,LIST,QSOJAC,RANGE,TABLES,LIMITS,PARAMS             OUTS0090
      EQUIVALENCE (KKARD(409),PRTDAT),(KKARD(377),LIST),                OUTS0100
     * (KKARD(401),QSOJAC),(KKARD(393),RANGE),                          OUTS0110
     * (KKARD(361),TABLES),(KKARD(337),LIMITS),(KKARD(345),PARAMS)      OUTS0120
C                                                                       OUTS0130
C  FUNCTION TABLES, PARAMETERS AND VALUE STORAGE                        OUTS0140
      COMMON /PLOTCM/ IRPG,IPNG,IPH,PPH,PPH2,IVC,IVA,IVF,IPOSOP(2),     OUTS0150
     * NSETS,IPORD,                                                     OUTS0160
     * TTLORD(12),TTLFRS(12),NPLOTS,NX,NY,XMIN,YMIN,DX,DY,NZ,CH(55),    OUTS0170
     * RECT(81,41),ITF(20),ISF(20),ITYF(20),IVPF(20),NF(20),F(14,20),   OUTS0180
     * ITTLF(12,20),KTF(20),PZMIN(20),PZMAX(20),XTB(14,10),YTB(14,10),  OUTS0190
     * C(7,10),NTB(10),KTB(10),OUT(20,36,8),ITTLP(12,8),KTPP(8),OLP(8), OUTS0200
     * OHP(8),ISP(8),ITP(8),PZMIN(8),PZMAX(8),ITYP(8),ITYS(8),IVPF(8)   OUTS0210
      INTEGER CH                                                        OUTS0220
C                                                                       OUTS0230
      INTEGER TXEXT(2),TXINT(2),TXYLI(2),TXYLL(2),TXFLL(2)              OUTS0240
      DATA MFCNS/20/,MF/14/                                             OUTS0250
      DATA TXEXT,TXINT,TXX,TXY/6HEXTERN,2HAL,6HINTERN,2HAL,1HX,1HY/     OUTS0260
      DATA TXXLL,TXYLL,TXFLL/5HX-LIN,5HX-LOG,5HY-LIN,5HY-LOG,6HLINEAR,  OUTS0270
     * 6HLOG   /                                                        OUTS0280
C                                                                       OUTS0290
      IF(RANGE) GO TO 120                                               OUTS0300
      IF(QSOJAC) GO TO 140                                              OUTS0310
      IF(TABLES) WHY(16)=.FALSE.                                        OUTS0320
      IF(LIMITS) WHY(17)=.FALSE.                                        OUTS0330
      IF(PARAMS) WHY(18)=.FALSE.                                        OUTS0340
      L = 0                                                             OUTS0350
C                                                                       OUTS0360
C  READ FUNCTION DEFINITIONS (TABLES, LIMITS, PARAMS)                   OUTS0370
C                                                                       OUTS0380
  300 READ (5,301) IT,IS,IVP,IDUM,JDUM,KDUM,ADUM,BDUM,CDUM,DDUM,EDUM    OUTS0390
  301 FORMAT(2I1,1X,A1,1X,3I5,5E10.0)                                   OUTS0400
```

```
      IF(IT.EQ.0 .AND. IS.EQ.0) GO TO 100                               OUTS0410
      IF (IT.LE.0) IT=0                                                 OUTS0420
      IF (IS.LE.0) IS=0                                                 OUTS0430
      L = L+1                                                           OUTS0440
      ITY = IT*10+IS                                                    OUTS0450
      IS = IS+1                                                         OUTS0460
      IF (IT.GT.0) GO TO 302                                            OUTS0470
      IT = IS+10                                                        OUTS0480
      IS = IT                                                           OUTS0490
  302 IF (TABLES) GO TO 320                                             OUTS0500
      IF(LIMITS) GO TO 340                                              OUTS0510
      IF(PARAMS) GO TO 360                                              OUTS0520
      IF(.NOT.OCCPRT) WRITE(6,FCC) PC,KTCD,KARD                         OUTS0530
      OCCPRT = .TRUE.                                                   OUTS0540
      WRITE(6,303)                                                      OUTS0550
  303 FORMAT(46H0******  REQUIRED FIELD(S) MISSING OR IN ERROR)         OUTS0560
      CALL FLUSH                                                        OUTS0570
      DO 306 I=16,20                                                    OUTS0580
  306 WHY(I) = .TRUE.                                                   OUTS0590
      GO TO 100                                                         OUTS0600
C                                                                       OUTS0610
C  READ TABLES AND/OR THOR COEFS.                                       OUTS0620
C                                                                       OUTS0630
  320 IT = (IT+2)/3                                                     OUTS0640
      IF(IT.GT.0 .AND. IT.LT.3) GO TO 323                               OUTS0650
      IF(.NOT.OCCPRT) WRITE(6,FCC) PC,KTCD,KARD                         OUTS0660
      OCCPRT = .TRUE.                                                   OUTS0670
      WRITE(6,324)                                                      OUTS0680
  324 FORMAT(47H0******  ILLEGAL VALUE IN DATA -- FUNCTION CODE)        OUTS0690
      CALL FLUSH                                                        OUTS0700
      WHY(16) = .TRUE.                                                  OUTS0710
      GO TO 100                                                         OUTS0720
  323 IF(IT.EQ.1) GO TO 322                                             OUTS0730
      READ(5,321) (C(I,IS),I=1,7)                                       OUTS0740
  321 FORMAT(7F10.0)                                                    OUTS0750
      IF (PRTDAT) WRITE(6,325) IS,TXEXT,(C(I,IS),I=1,7)                 OUTS0760
  325 FORMAT(1H0,18X,16HTHOR COEF. SET NO.,I3,2H (,A6,A2,9H FORM)       OUTS0770
     * 7F10.4)                                                          OUTS0780
      C(3,IS) = -0.666667*C(3,IS)                                       OUTS0790
      C(4,IS) = -C(3,IS)-C(4,IS)                                        OUTS0800
```

```
      C(5,IS) = -C(5,I')                                                OUTS0810
      IF (LIST) WRITE(6,325) IS,TXPAT,(C(I,IS),I=1,7)                   OUTS0820
      GO TO 300                                                         OUTS0830
  322 READ(5,321) (XTR(I,IS),I=1,IDUM)                                  OUTS0840
      NTR(IS) = IDUM                                                    OUTS0850
      KTR(IS) = JDUM                                                    OUTS0860
      READ(5,321) (YTR(I,IS),I=1,IDUM)                                  OUTS0870
      II = IS-1                                                         OUTS0880
      IX = JDUM/10+1                                                    OUTS0890
      IY = MOD(JDUM,10)+1                                               OUTS0900
      IF(.NOT.PRTDAT) GO TO 328                                         OUTS0910
      WRITE (6,327) II,IDUM,TXXLL(IX),TXYLL(IY)                         OUTS0920
  327 FORMAT(1H0,18X,9HTABLE NO.,I3,I7,7H VALUES,5X,A5,1X,A5)           OUTS0930
      WRITE(6,326) TXX,TXEXT,(XTR(I,IS),I=1,IDUM)                       OUTS0940
  326 FORMAT(19X,A1,2H VALUES (,A6,A2,6H FORM) ,7E13.5/(45X,7E13.5))   OUTS0950
      WRITE(6,326) TXY,TYEXT,(YTR(I,IS),I=1,IDUM)                       OUTS0960
  328 IF (IX.EQ.1) GO TO 346                                            OUTS0970
      DO 345 I=1,IDUM                                                   OUTS0980
  345 XTR(I,IS) = ALOG10(XTR(I,IS))                                     OUTS0990
  346 IF (IY.EQ.1) GO TO 348                                            OUTS1000
      DO 347 I=1,IDUM                                                   OUTS1010
  347 YTR(I,IS) = ALOG10(YTR(I,IS))                                     OUTS1020
  348 CONTINUE                                                          OUTS1030
      IF(.NOT.LIST) GO TO 300                                           OUTS1040
      IF (.NOT.PRTDAT) WRITE (6,327) II,IDUM,TXXLL(IX),TXYLL(IY)        OUTS1050
      WRITE(6,326) TXX,TXINT,(XTR(I,IS),I=1,IDUM)                       OUTS1060
      WRITE(6,326) TXY,TXINT,(YTR(I,IS),I=1,IDUM)                       OUTS1070
      GO TO 300                                                         OUTS1080
C                                                                       OUTS1090
C  READ OUTPUT FCN TYPES (LOG OR LINEAR) AND BOUNDS (ALWAYS LINEAR)     OUTS1100
C                                                                       OUTS1110
  340 KTP(IT) = JDUM                                                    OUTS1120
      IX = JDUM/10                                                      OUTS1130
      IY = MOD(JDUM,10)+1                                               OUTS1140
      IF (PRTDAT) WRITE (6,342) IT,TYEXT,TXFLL(IY),IX,ADUM,BDUM         OUTS1150
  342 FORMAT(1H0,18X,8HFCN. NO.,I3,2H (,A6,A2,26H FORM)     OUTPUT MODE OUTS1160
     =IS ,A6,5X,17HSMOOTHING ORDER =,I2/19X,13HPLOT BOUNDS =,E12.4,     OUTS1170
     * 3H TO,E12.4)                                                     OUTS1180
      IF (IY.EQ.1 .OR. ADUM.GE.BDUM) GO TO 341                          OUTS1190
      ADUM = ALOG10(ADUM)                                               OUTS1200
```

```
      BDUM = ALOG10(BDUM)                                               OUTS1210
  341 OZMIN(IT) = ADUM                                                  OUTS1220
      OZMAX(IT) = BDUM                                                  OUTS1230
      IF (LIST) WRITE (6,342) IT,TYPIT,TXFLL(IY),IY,ADUM,BDUM           OUTS1240
      GO TO 300                                                         OUTS1250
C                                                                       OUTS1260
C  READ OUTPUT FCN TITLES AND PARAMETERS                                OUTS1270
C                                                                       OUTS1280
  360 IF(L.LE.MFCNS) GO TO 361                                          OUTS1290
      WHY(18) = .TRUE.                                                  OUTS1300
  363 IF(.NOT.OCCPRT) WRITE(6,FCC) PC,KTCD,KARD                         OUTS1310
      OCCPRT = .TRUE.                                                   OUTS1320
      WRITE(6,364)                                                      OUTS1330
  364 FORMAT(42H0******  TOO MANY SETS OF OUTPUT FUNCTIONS)             OUTS1340
      CALL FLUSH                                                        OUTS1350
      GO TO 100                                                         OUTS1360
  361 ITF(L) = IT                                                       OUTS1370
      ISF(L) = IS                                                       OUTS1380
      ITYF(L) = ITY                                                     OUTS1390
      IF (IVP.EQ.PS) IVP=PP                                             OUTS1400
      IVPF(L) = IVP                                                     OUTS1410
      NF(L) = IDUM                                                      OUTS1420
      NSETS = L                                                         OUTS1430
      READ(5,362) (ITTLF(I,L),I=1,12)                                   OUTS1440
  362 FORMAT(12A6)                                                      OUTS1450
      IF (PRTDAT.OR.LIST) WRITE(6,365) L,ITY,IVP,IDUM,                  OUTS1460
     * (ITTLF(I,L),I=1,12)                                              OUTS1470
  365 FORMAT(1H0,18X16HOUTPUT DECL. NO.,I3,10H, FCN. NO.,I3,1H-,A1,     OUTS1480
     * 1H,,I6,12H PARAMETERS./19X,8HTITLE = ,12A6)                      OUTS1490
      DO 367 I=1,MF                                                     OUTS1500
  367 F(I,L) = 0.                                                       OUTS1510
      IF(IDUM.LE.0) GO TO 300                                           OUTS1520
      READ(5,321) (F(I,L),I=1,IDUM)                                     OUTS1530
      IF (PRTDAT.OR.LIST) WRITE(6,366) (F(I,L),I=1,IDUM)                OUTS1540
  366 FORMAT(19X,12HPARAMETERS =,7E13.5/(31X,7E13.5))                   OUTS1550
      GO TO 300                                                         OUTS1560
C                                                                       OUTS1570
C  READ NEW SET OF PLOT RANGE PARAMETERS                                OUTS1580
C                                                                       OUTS1590
  120 READ(5,121) NX,NY,XMIN,YMIN,DX,DY                                 OUTS1600
```

```
121 FORMAT(5X,2I5,5X,5F10.2)                                           OUTS1610
    IF (PRTDAT.OR.LIST) WRITE(6,122) NX,NY,XMIN,YMIN,DX,DY             OUTS1620
122 FORMAT(1H0,18X4HNX =,I3,5X,4HNY =,I3,5X,6HXMIN =,E12.4,5X,6HYMIN =OUTS1630
   *,E12.4,5X,4HDX =,E12.4,5X,4HDY =,E12.4)                           OUTS1640
    GO TO 100                                                          OUTS1650
C                                                                      OUTS1660
C  READ NEW SET OF SOURCE PLOT CHARACTERS, AND SET NO. CONTOUR INTERVALS.OUTS1670
C                                                                      OUTS1680
140 READ(5,141) NZ,(CH(I),I=1, Z),CH(NZ+1),CH(NZ+2),CH(NZ+3),CH(NZ+4) OUTS1690
   *,CH(NZ+5)                                                          OUTS1700
141 FORMAT(5X,I5,5X,15A1)                                              OUTS1710
    IDUM = NZ+4                                                        OUTS1720
    IF(LIST.OR.PRTDAT) WRITE(6,142) NZ,((CH(I),J=1,3),I=1,4),         OUTS1730
   * (CH(NZ+5),J=1,3),((CH(I),J=1,3),I=5,IDUM)                         OUTS1740
142 FORMAT(1H0,10XI2,32H CONTOUR INTERVALS.  BORDER CHARS. ARE ,3A1,  OUTS1750
   * 5H AND ,3A1,11H, UNDEF. = ,3A1,8H, LOW = ,3A1,9H, HIGH = ,3A1/   OUTS1760
   * 19X,10HIN-RANGE =,20(1X,3A1,1H,)/(29X,20(1X,3A1,1H,)))           OUTS1770
100 RETURN                                                             OUTS1780
    END                                                                OUTS1790
```

```
      SUBROUTINE COMPUT                                                 COMP0010
C                                                                       COMP0020
C  OUTPUT FUNCTION LOOP                                                 COMP0030
C  OUT(K,IPH,LL) IS A FUNCTION OF LL. PACK(K,IMO), PACK(K,IMO+1),       COMP0040
C  PACK(K,IMO+2), AMORD(IM), ETC., FOR ALL IDA AND IM.                  COMP0050
C                                                                       COMP0060
C  CONTROL CARD TABLES AND GENERAL PROGRAM CONTROL VARIABLES            COMP0070
      COMMON /CNTRL/ GOCRD,GOCPRT,KTCD,FCC(8),KARD(14),PC,PS,PD,PP,     COMP0080
     * TXTIME(2),TXDATE(2),WHY(20),KKARD(800)                           COMP0090
      LOGICAL GOCRD,GOCPRT,WHY                                          COMP0100
      INTEGER TXTIME,TXDATE,FCC,PC,PS,PD,PP                             COMP0110
      LOGICAL PRTDAT,LIST,PLOT,GWKPLT                                   COMP0120
      EQUIVALENCE (KKARD(489),PRTDAT),(KKARD(377),LIST),                COMP0130
     * (KKARD(369),PLOT),(KKARD(353),GWKPLT)                            COMP0140
C                                                                       COMP0150
C  FUNCTION TABLES, PARAMETERS AND VALUE STORAGE                        COMP0160
      COMMON /PLOTCM/ NRNG,DRNG,NPH,DPH,DPH2,IVO,IVA,IVE,IPOSOP(2),     COMP0170
     * NSETS,IDORD,                                                     COMP0180
     * TTLORD(12),TTLFRG(12),NPLOTS,NX,NY,XMIN,YMIN,DX,DY,NZ,CH(55),    COMP0190
     * RECT(81,41),ITF(20),ISF(20),ITYF(20),IVFF(20),NF(20),F(14,20),   COMP0200
     * ITTLF(12,20),KTF(20),PZMIN(20),PZMAX(20),XTB(14,10),YTB(14,10),  COMP0210
     * C(7,10),NTB(10),KTB(10),OUT(20,36,8),ITTLP(12,8),KTPP(8),CLP(8), COMP0220
     * OHP(8),ISP(8),ITP(8),PZMIN(8),PZMAX(8),ITYP(8),ITYS(8),IVPF(8)   COMP0230
C                                                                       COMP0240
      DIMENSION PACK(20,180),KPK1(60),KPK2(60)                          COMP0250
      DIMENSION SCA(2),AMORD(30)                                        COMP0260
      LOGICAL GTWO                                                      COMP0270
C                                                                       COMP0280
      DATA PI,ELN10,RADEG/3.14159265,2.30258509,57.2957795/             COMP0290
      DATA MPLOTS/8/,IFRTAP/4/                                          COMP0300
C     CONKE    KINETIC ENERGY CONVERSION FACTOR, =2*7000*32.17          COMP0310
C     CONM     GRAINS/LB, =7000                                         COMP0320
C     CONMV    MOMENTUM CONVERSION FACTOR, =7000*32.17                  COMP0330
      DATA CONKE,CONM/450380.,7000./,CONMV/225190./                     COMP0340
C                                                                       COMP0350
C  SELECT FUNCTIONS TO BE EVALUATED, AND PLOT                           COMP0360
  110 READ(5,111) (ITYS(LP),LP=1,MPLOTS)                                COMP0370
  111 FORMAT(14(I2,3X))                                                 COMP0380
      WHY(5) = .FALSE.                                                  COMP0390
      DO 112 NS=1,MPLOTS                                                COMP0400
```

```
      IF(ITYS(NS).LE.0) GO TO 113                                       COMP0410
  112 CONTINUE                                                          COMP0420
      NS = NPLOTS+1                                                     COMP0430
  113 NS = NS-1                                                         COMP0440
      IF (LIST.OR.PRTOUT) WRITE(6,116) NS,(ITYS(I),I=1,NS)              COMP0450
  116 FORMAT(1H0,13X12,16H PLOT REQUESTS../(29X,14I5))                  COMP0460
      IF (NS.GT.0) GO TO 119                                            COMP0470
      IF(.NOT.OCCPRT) WRITE(6,500) PC,KTCD,KARD                         COMP0480
      OCCPRT = .TRUE.                                                   COMP0490
      WRITE(6,115)                                                      COMP0500
  115 FORMAT(30H0****** NO FUNCTIONS SELECTED)                          COMP0510
      GO TO 117                                                         COMP0520
  119 IF(PLOT.OR.QWKPLT) GO TO 114                                      COMP0530
      IF(.NOT.OCCPRT) WRITE(6,500) PC,KTCD,KARD                         COMP0540
      OCCPRT = .TRUE.                                                   COMP0550
      WRITE(6,118)                                                      COMP0560
  118 FORMAT(47H0******  NEITHER 'PLOT' NOR 'KWIKPLOT' SELECTED)        COMP0570
  117 WHY(5) = .TRUE.                                                   COMP0580
      GO TO 100                                                         COMP0590
C  RETURN IF ERROR FLAGS ARE SET                                        COMP0600
  114 IF(WHY(1).OR.WHY(5).OR.WHY(6)) GO TO 100                          COMP0610
C  READ HEADER INFO FOR CURRENT FRAGMENT FIELD                          COMP0620
      READ (IFRTAP) JHDR,IPOSOP                                         COMP0630
      READ (IFRTAP) TTLORD,TTLFRG,NM,XK,NRNG,DRNG,NPH                   COMP0640
      READ (IFRTAP) (AMORD(I),I=1,NM)                                   COMP0650
      I = INTI(IPOSOP,1,2,0,IDORD)                                      COMP0660
      IVO = PS                                                          COMP0670
      IVA = PS                                                          COMP0680
      IVF = PS                                                          COMP0690
      CALL MOVE(IVO,1,3,IPOSOP,3)                                       COMP0700
      CALL MOVE(IVA,1,3,IPOSOP,4)                                       COMP0710
      CALL MOVE(IVF,1,3,IPOSOP,5)                                       COMP0720
      IMX = 6*NM                                                        COMP0730
      IMY = 2*NM                                                        COMP0740
      DPH = PI/FLOAT(NPH)                                               COMP0750
      DPH2 = DPH/2.                                                     COMP0760
      PHI = DPH2                                                        COMP0770
C  PROCESS FRAGMENT FIELD AN AZIMUTH RAY AT A TIME                      COMP0780
      DO 510 IPH=1,NPH                                                  COMP0790
      READ (IFRTAP) (KPK1(J),KPK2(J),J=1,IMY),                          COMP0800
```

```
     * ((PACK(I,J),I=1,NPLO),J=1,I X)                                  COMP0810
      DO 515 L=1,NPLOTS                                                COMP0820
      DO 515 K=1,NPLO                                                  COMP0830
  515 OUT(K,IPH,L) = 1.E-30                                            COMP0840
      LL = 0                                                           COMP0850
      DO 521 LP=1,NS                                                   COMP0860
      IK = 1                                                           COMP0870
      ITY = ITYS(LP)                                                   COMP0880
      IF(MOD(ITY,30).GT.[illegible] .OR. ITY. [illegible]) GO TO 560   COMP0890
      ITY = ITY-20                                                     COMP0900
      IK = 2                                                           COMP0910
  560 DO 550 L=1,NSETS                                                 COMP0920
      IF(ITY.EQ.ITYPE(L)) GO TO 551                                    COMP0930
  550 CONTINUE                                                         COMP0940
      IF(.NOT.OCCPRT) WRITE(6,FCO) [illegible],KTCD,KARD               COMP0950
      OCCPRT = .TRUE.                                                  COMP0960
      IF (IPH.GT.1) GO TO 521                                          COMP0970
      WRITE(6,552) ITYS(LP)                                            COMP0980
  552 FORMAT(1H0*****  PLOT CODE,I3,32H NOT DEFINED)                   COMP0990
      GO TO 521                                                        COMP1000
  551 IT = ITF(L)                                                      COMP1010
      IS = ISF(L)                                                      COMP1020
      IVP = IVPF(L)                                                    COMP1030
      STWO = IK.EQ.2 .OR. MOD(ITYS(LP)/10,3).EQ.2                      COMP1040
      DO 553 JK=1,IK                                                   COMP1050
      LL = LL+1                                                        COMP1060
      IF(LL.LE.NPLOTS) GO TO 512                                       COMP1070
      WHY(5) = .TRUE.                                                  COMP1080
      GO TO 363                                                        COMP1090
  512 IL = LL-1+IK                                                     COMP1100
      IF(IPH.GT.1) GO TO 554                                           COMP1110
      KTPP(LL) = KTP(IT)                                               COMP1120
      OLP(LL) = OZMIN(IT)                                              COMP1130
      OHP(LL) = OZMAX(IT)                                              COMP1140
      IF (OZMIN(IT).LT.OZMAX(IT)) GO TO 563                            COMP1150
      OLP(LL) = -1.E+30                                                COMP1160
      IF (MOD(KTP(IT),10).GT.0) OLP(LL)=1.E-30                         COMP1170
      OHP(LL) = 1.E+30                                                 COMP1180
  563 PZMIN(LL) = OZMIN(IT)                                            COMP1190
      PZMAX(LL) = OZMAX(IT)                                            COMP1200
```

```
      ITP(LL) = IT                                                      COMP1210
      ISP(LL) = IS                                                      COMP1220
      ITYP(LL) = ITY                                                    COMP1230
      IVPP(LL) = IVP                                                    COMP1240
      DO 556 I=1,12                                                     COMP1250
  556 ITTLP(I,LL) = ITTL(I,I)                                           COMP1260
  554 IF(JK.EQ.2) GO TO 555                                             COMP1270
      IMO = 1                                                           COMP1280
      JM = 0                                                            COMP1290
      DO 511 IM=1,NM                                                    COMP1300
      W = AMORD(IM)                                                     COMP1310
C  LOOK UP THRESHOLD VELOCITY IF USING A DAMAGE THRESHOLD TABLE         COMP1320
      IF(IT.NE.1 .AND. IT.NE.2) GO TO 557                               COMP1330
      DUM = W                                                           COMP1340
      IF (KTB(IS).GE.100) DUM=ALOG10(DUM)                               COMP1350
      IDUM = NTB(IS)                                                    COMP1360
      DO 558 I=2,IDUM                                                   COMP1370
      IF(DUM.LE.XTB(I,IS)) GO TO 559                                    COMP1380
  558 CONTINUE                                                          COMP1390
      I = IDUM                                                          COMP1400
  559 VDUM = YTB(I-1,IS)+(YTB(I,IS)-YTB(I-1,IS))*(DUM-XTB(I-1,IS))/     COMP1410
     * (XTB(I,IS)-XTB(I-1,IS))                                          COMP1420
      IF (MOD(KTB(IS),100).GT.0) VDUM=EXP(VDUM*ELN10)                   COMP1430
  557 DO 519 IDA=1,2                                                    COMP1440
      JM = JM+1                                                         COMP1450
      K1 = KPK1(JM)                                                     COMP1460
      IF(K1.LE.0) GO TO 519                                             COMP1470
      K2 = KPK2(JM)                                                     COMP1480
      DO 520 K=K1,K2                                                    COMP1490
      V = PACK(K,IMO)                                                   COMP1500
      A = PACK(K,IMO+1)                                                 COMP1510
      SCA(1) = ABS(SIN(A))                                              COMP1520
      SCA(2) = ABS(COS(A))                                              COMP1530
      D = PACK(K,IMO+2)                                                 COMP1540
C  ELIMINATE CONSIDERATION OF FRAGMENTS FALLING BELOW A PRIORI T.HOLDS  COMP1550
      IF (D.LE.0.) GO TO 520                                            COMP1560
      IF (W.LT.F(1,L) .OR. V.LT.F(2,L)) GO TO 520                       COMP1570
      FMV = W*V/CONMV                                                   COMP1580
      FKE = W*V*V/CONKE                                                 COMP1590
      IF (FMV.LT.F(3,L) .OR. FKE.LT.F(4,L)) GO TO 520                   COMP1600
```

```
      GO TO (210,210,220,220,220,220,200,200,200,200,200,260,265,270,      COMP1610
     * 275,290,290,290,295,295),IT                                         COMP1620
  200 IF(IPH.GT.1) GO TO 521                                               COMP1630
      IF(.NOT.OCCPRT) WRITE(6,FCC) PC,KTCD,KARD                            COMP1640
      OCCPRT = .TRUE.                                                      COMP1650
      WRITE(6,201) ITYP(LL)                                                COMP1660
  201 FORMAT(22H0******  FUNCTION TYPE,I3,14H HAS NO DRIVER)               COMP1670
      GO TO 521                                                            COMP1680
C                                                                          COMP1690
C  OUTPUT FUNCTIONS                                                        COMP1700
C                                                                          COMP1710
C  FUNCTION 01 -- NUMBER OF FRAGMENTS PER SQ FT                            COMP1720
  260 DUM = D*SCA(1)                                                       COMP1730
      GO TO 209                                                            COMP1740
C  FUNCTION 02 -- FRAGMENT WEIGHT (LB) PER SQ FT                           COMP1750
  265 DUM = D*SCA(1)*W/CONV                                                COMP1760
      GO TO 209                                                            COMP1770
C  FUNCTION 03 -- FRAGMENT MOMENTUM (LB-SEC) PER SQ FT                     COMP1780
  270 DUM = D*SCA(1)*FMV                                                   COMP1790
      GO TO 209                                                            COMP1800
C  FUNCTION 04 -- FRAGMENT K.E. (FT-LB) PER SQ FT                          COMP1810
  275 DUM = D*SCA(1)*FKE                                                   COMP1820
      GO TO 209                                                            COMP1830
C  FUNCTION 08 AND 09 -- MUST BE AT END OF EXEC LIST.                      COMP1840
C  LIST MUST IMPLY FEWER THAN MPLOTS PLOTS.                                COMP1850
C  PRODUCE PLOT OF AVG VELOCITY AND OF AVG STRIKING ANGLE                  COMP1860
  290 OUT(K,IPH,LL) = OUT(K,IPH,LL)+V*D*W                                  COMP1870
      GO TO 520                                                            COMP1880
  295 OUT(K,IPH,LL) = OUT(K,IPH,LL)+A*RADEG*D*W                            COMP1890
      OUT(K,IPH,LL+1) = OUT(K,IPH,LL+1)+D*W                                COMP1900
      GO TO 520                                                            COMP1910
C  FUNCTION 1X -- TOTAL NO. LETHAL FRAGMENTS (SEE TABLE NO. X)             COMP1920
C  FUNCTION 2X -- PROB. OF KILL                (SEE TABLE NO. X)           COMP1930
C  FUNCTION 3X -- (DO BOTH 1X AND 2X)                                      COMP1940
C  (FIRST, FIND VELOCITY RELATIVE TO (MOVING) TARGET)                      COMP1950
  210 IF (V.LE.VDUM) GO TO 520                                             COMP1960
      DUM = D                                                              COMP1970
      IF(.NOT.QTWO) GO TO 209                                              COMP1980
      ADUM = EXP(-D*(F(8,L)*SCA(1)+F(9,L)*SCA(2)))                         COMP1990
      IF(IT.EQ.2) GO TO 207                                                COMP2000
```

```
      GO TO 208                                                         COMP2010
C  FUNCTION 4X -- TOTAL NO. LETHAL FRAGMENTS (SEE THOR COEF. SET 10, X) COMP2020
C  FUNCTION 5X -- PROB. OF KILL                (SEE THOR COEF. SET 10, X) COMP2030
C  FUNCTION 6X -- (USE BOTH 4X AND 5X)                                  COMP2040
  220 PDUM = 1.                                                         COMP2050
      IF (F(11,L).GT.0.) PDUM=F(11,L)**C(7,IS)                          COMP2060
      CDUM = 10.**C(1,IS)*F(10,L)**C(2,IS)*XK**C(3,IS)*...**C(4,IS)     COMP2070
     * *V**C(6,IS)*PDUM                                                 COMP2080
      DO 221 I=1,2                                                      COMP2090
      IF (V.GT.SCA(I)*C(5,IS)*...) GO TO 222                            COMP2100
  221 CONTINUE                                                          COMP2110
      GO TO 520                                                         COMP2120
  222 DUM = 0                                                           COMP2130
      IF(.NOT.CT(40)) GO TO 209                                         COMP2140
      ADUM = EXP(-D*(F(9,L)*SCA(1)-F(9,L)*SCA(2)))                      COMP2150
      IF(IT.EQ.5) GO TO 207                                             COMP2160
      GO TO 208                                                         COMP2170
C  CONTRIBUTE TO OUT(K,IPH,LL) AND/OR OUT(K,IPH,LL)                     COMP2180
C  DUM IS A 'PARTIAL NUMBER DENSITY'                                    COMP2190
C  ADUM IS A 'PARTIAL PROBABILITY'                                      COMP2200
  209 OUT(K,IPH,LL) = OUT(K,IPH,LL)+DUM                                 COMP2210
      GO TO 520                                                         COMP2220
  208 OUT(K,IPH,LL) = OUT(K,IPH,LL)+DUM                                 COMP2230
  207 OUT(K,IPH,LL) = 1.-(1.-OUT(K,IPH,LL))*ADUM                        COMP2240
  520 CONTINUE                                                          COMP2250
  519 IMO = IMO+3                                                       COMP2260
  511 CONTINUE                                                          COMP2270
  555 IT = IT+1                                                         COMP2280
      L = L+1                                                           COMP2290
  553 ITY = ITY+10                                                      COMP2300
  521 CONTINUE                                                          COMP2310
      NPLOTS = LL                                                       COMP2320
C  SCALE FUNCTION VALUES, TAKE LOG IF NEEDED.                           COMP2330
      DO 523 LL=1,NPLOTS                                                COMP2340
      IS = ISP(LL)                                                      COMP2350
      IT = ITP(LL)                                                      COMP2360
      DO 522 K=1,NRNG                                                   COMP2370
      DUM = OUT(K,IPH,LL)                                               COMP2380
      IF (ABS(DUM).LE.1.E-38) GO TO 524                                 COMP2390
      IF(IT.NE.19 .AND. IT.NE.20) GO TO 525                             COMP2400
```

```
C  NORMALIZE BY DIVIDING BY FRAG. NO. DENSITY, FOR FCNS 08 AND 09 ONLY  COMP2410
      ILL = LL+1                                                         COMP2420
      IF(IT.EQ.19) ILL = LL+2                                            COMP2430
      DUM1 = OUT(K,IPH,ILL)                                              COMP2440
      IF(DUM1.NE.0.) DUM=DUM/DUM1                                        COMP2450
  525 IF (DUM.LT.OLP(LL)) DUM=OLP(LL)                                    COMP2460
      IF (DUM.GT.OHP(LL)) DUM=OHP(LL)                                    COMP2470
      IF (MOD(KTPP(LL),10).GT.0) DUM=ALOG10(DUM)                         COMP2480
      GO TO 522                                                          COMP2490
C  IF NO VALUE HAS BEEN STORED, SET VALUE UNDEFINED (=1.E+38)            COMP2500
  524 DUM = 1.E+38                                                       COMP2510
  522 OUT(K,IPH,LL) = DUM                                                COMP2520
  523 CONTINUE                                                           COMP2530
  510 CONTINUE                                                           COMP2540
      GO TO 100                                                          COMP2550
  363 IF(.NOT.QCCPRT) WRITE(6,FCC) PC,KTCD,KARD                          COMP2560
      QCCPRT = .TRUE.                                                    COMP2570
      WRITE(6,364)                                                       COMP2580
  364 FORMAT(42H0******  TOO MANY SETS OF OUTPUT FUNCTIONS)              COMP2590
      CALL FLUSH                                                         COMP2600
  100 RETURN                                                             COMP2610
      END                                                                COMP2620
```

```
      SUBROUTINE OUTPUT
C
C  GENERATE PLOTS USING PRINTER AND/OR WRITE FUNCTIONS TO TAPE
C
C  CONTROL CARD TABLES AND GENERAL PROGRAM CONTROL VARIABLES
      COMMON /CNTRL/ QCCRD,QCCPRT,KTCD,FCC(8),KARD(14),PC,PS,PD,PP,
     * TXTIME(2),TXDATE(2),WHY(20),KKARD(800)
      LOGICAL QCCRD,QCCPRT,WHY
      INTEGER TXTIME,TXDATE,FCC,PC,PS,PD,PP
      LOGICAL LIST,PLOT,QWKPLT
      EQUIVALENCE (KKARD(377),LIST),(KKARD(369),PLOT),
     * (KKARD(353),QWKPLT)
C
C  FUNCTION TABLES, PARAMETERS AND VALUE STORAGE
      COMMON /PLOTC/ NRNG,DRNG,MPH,DPH,DPH2,IV2,IV4,IVF,IPOSOP(2),
     * NSETS,IPOPD,
     * TTLORD(12),TTLFRG(12),NPLOTS,NX,NY,XMIN,YMIN,DX,DY,NZ,CH(55),
     * RECT(81,41),ITF(20),ISF(20),ITYF(20),IVPF(20),NF(20),F(14,20),
     * ITTLF(12,20),KTP(20),QZMIN(20),QZMAX(20),XTB(14,10),YTB(14,10),
     * C(7,10),NTB(10),KTB(10),OUT(20,36,8),ITTLP(12,8),KTPP(8),OLP(8),
     * OHP(8),ISP(8),ITP(8),PZMIN(8),PZMAX(8),ITYP(8),ITYS(8),IVPP(8)
      INTEGER CH
C
      EXTERNAL PURPLE
      DIMENSION TT(41,4),TEMP(6,4),KTEMP(6,4),TEMQ(6,4)
      EQUIVALENCE (TT(1,1),TEMP(1,1)),(TT(1,3),KTEMP(1,1)),
     * (TT(1,5),TEMQ(1,1))
      DATA MXRNG/20/
      DATA ELN10/2.30258509/
      DATA MXORDR/5/,IHDR/6H*HEAD*/
      DATA IPLTAP/2/
C
      DO 600 LL=1,NPLOTS
      IF(.NOT.PLOT) GO TO 604
      I = IFXO(IPOSOP,6,2,0,ITYP(LL))
      CALL MOVE(IPOSOP,8,1,IVPP(LL))
      WRITE (IPLTAP) IHDR,IPOSOP
      WRITE (IPLTAP) TTLORD
      WRITE (IPLTAP) TTLFRG
      WRITE (IPLTAP) (ITTLP(I,LL),I=1,12)
```

```
604 PC = PD                                                          OUTP0410
      IF(QWKPLT) PC=PP                                               OUTP0420
      IF (.NOT. (LIST.OR.QWKPLT)) GO TO 605                          OUTP0430
      WRITE (6,602) PC,IORD,IVC,TTLORD,IVA,IVF,TTLFRG,ITYP(LL),      OUTP0440
     * IVPP(LL),(ITTLP(I,LL),I=1,12)                                 OUTP0450
602 FORMAT(A1,18X,19HDATA I.D.         =,I5,1H-,A1,5X,12A6/          OUTP0460
     * 19X,19HFRAG I.D.         =,4X,A1,1H=,A1,5X,12A6/              OUTP0470
     * 19X,19HPLOT I.D.         =,I5,1H-,A1,5X,12A6)                 OUTP0480
      IF (PLOT) WRITE(6,603) IPOSOP                                  OUTP0490
603 FORMAT(20H PLOT TAPE KEY      =,1X,A6,A2)                        OUTP0500
      WRITE (6,607)                                                  OUTP0510
607 FORMAT(1X)                                                       OUTP0520
C   FCN. VALUES COMPUTED ON POLAR GRID.  RE-EXPRESS ON A RECT. GRID. OUTP0530
605 ZMIN = 1.E38                                                     OUTP0540
      ZMAX = -1.E38                                                  OUTP0550
      DO 606 I=1,NRNG                                                OUTP0560
      DO 606 J=1,NPH                                                 OUTP0570
      IF (OUT(I,J,LL).GE.1.E+38) GO TO 606                           OUTP0580
      ZMIN = AMIN1(ZMIN,OUT(I,J,LL))                                 OUTP0590
      ZMAX = AMAX1(ZMAX,OUT(I,J,LL))                                 OUTP0600
606 CONTINUE                                                         OUTP0610
      Y = YMIN                                                       OUTP0620
      DO 610 IY=1,NY                                                 OUTP0630
      X = XMIN                                                       OUTP0640
      DO 611 IX=1,NX                                                 OUTP0650
      RECT(IX,IY) =                                                  OUTP0660
     * POLCAR(OUT(1,1,LL),MXRNG,NRNG,NPH,DRNG,DPH,ORNG,OPH2,X,Y)     OUTP0670
611 X = X+DX                                                         OUTP0680
610 Y = Y+DY                                                         OUTP0690
C   OPTIONALLY SMOOTH GRID.  (RECT(I,J)=AVG OF IT AND MANY NEIGHBORS) OUTP0700
      IORDER = KTPP(LL)/10                                           OUTP0710
      IOR = MINO(MXORDR,IORDER)                                      OUTP0720
      IF(IOR.LE.0) GO TO 613                                         OUTP0730
      IOR1 = IOR+1                                                   OUTP0740
      IDUM = NX+IOR1                                                 OUTP0750
      II = 0                                                         OUTP0760
      IXOR = 1-IOR1                                                  OUTP0770
      DO 640 IX=1,IDUM                                               OUTP0780
      DO 641 IY=1,NY                                                 OUTP0790
      II = II+1                                                      OUTP0800
```

```
      IF(IX.GT.IOR1) RECT(IXOR,IY)=TT(IY,II)                            OUTP0810
      IF(IX.GT.NX) GO TO 641                                            OUTP0820
      IL = MAX0(1,IX-IOR)                                               OUTP0830
      IH = MIN0(NX,IX+IOR)                                              OUTP0840
      JL = MAX0(1,IY-IOR)                                               OUTP0850
      JH = MIN0(NY,IY+IOR)                                              OUTP0860
      DUM = 0.                                                          OUTP0870
      DO 642 I=IL,IH                                                    OUTP0880
      DO 643 J=JL,JH                                                    OUTP0890
  644 DUM = DUM+RECT(I,J)                                               OUTP0900
  643 CONTINUE                                                          OUTP0910
  642 CONTINUE                                                          OUTP0920
      TT(IY,II) = DUM/FLOAT((IH-IL+1)*(JH-JL+1))                        OUTP0930
  641 CONTINUE                                                          OUTP0940
  640 IXOR = IXOR+1                                                     OUTP0950
  613 IF (PZMIN(LL).GE.PZMAX(LL)) GO TO 612                             OUTP0960
      ZMIN = PZMIN(LL)                                                  OUTP0970
      ZMAX = PZMAX(LL)                                                  OUTP0980
  612 CONTINUE                                                          OUTP0990
C  IF REQUESTED, OUTPUT FUNCTION IN POLAR AND RECT. FORM                OUTP1000
      IF (.NOT.PLOT) GO TO 645                                          OUTP1010
      WRITE (IPLTAP) NRNG,NPH,DRNG,DPH2,DRNG,DPH,ZMIN,ZMAX              OUTP1020
      WRITE (IPLTAP) ((OUT(I,J,LL),I=1,NRNG),J=1,NPH)                   OUTP1030
      WRITE (IPLTAP) NX,NY,XMIN,YMIN,DX,DY,ZMIN,ZMAX                    OUTP1040
      WRITE (IPLTAP) ((RECT(I,J),I=1,NX),J=1,NY)                        OUTP1050
C  GEN. PRINTER CONTOUR PLOT OF FCN. (IF QWKPLT=T)                      OUTP1060
  645 IF (.NOT.QWKPLT) GO TO 600                                        OUTP1070
      CALL SOJAC(PURPLF,YMIN,YMIN+FLOAT(NY-1)*DY,XMIN,XMIN+FLOAT(NX-1)* OUTP1080
     * DX,ZMIN,ZMAX,NX,1,NZ,CH)                                         OUTP1090
      DZ = (ZMAX-ZMIN)/FLOAT(NZ)                                        OUTP1100
      Z = ZMIN                                                          OUTP1110
      NL = (NZ+5)/4                                                     OUTP1120
      NI = NL-(NL*4-(NZ+2))                                             OUTP1130
      K = 3                                                             OUTP1140
      DO 682 J=1,4                                                      OUTP1150
      DO 680 I=1,NL                                                     OUTP1160
      K = K+1                                                           OUTP1170
      TEMP(I,J) = Z                                                     OUTP1180
      IF (MOD(KTPP(LL),10).GT.0) TEMP(I,J)=EXP(TEMP(I,J)*ELN10)         OUTP1190
      KTEMP(I,J) = CH(K)                                                OUTP1200
```

```
680 Z = Z+DZ                                                         OUTP1210
682 CONTINUE                                                         OUTP1220
    NJ = NL-1                                                        OUTP1230
    DO 684 J=1,4                                                     OUTP1240
    IF(NJ.LE.0) GO TO 687                                            OUTP1250
    DO 686 I=1,NJ                                                    OUTP1260
686 TEMQ(I+1,J) = TEMP(I,J)                                          OUTP1270
687 IF(J.EQ.4) GO TO 684                                             OUTP1280
    TEMQ(1,J+1) = TEMP(NL,J)                                         OUTP1290
684 CONTINUE                                                         OUTP1300
    TEMQ(1,1) = -1.E38                                               OUTP1310
    TEMP(NI,4) = +1.E38                                              OUTP1320
    NJ = 4                                                           OUTP1330
    WRITE(6,685)                                                     OUTP1340
685 FORMAT(//57X,19HTHUS SCALES SOJAC..//)                           OUTP1350
    DO 683 I=1,NL                                                    OUTP1360
    IF(I.GT.NI) NJ=3                                                 OUTP1370
    WRITE (6,681) (TEMQ(I,J),(KTEMP(I,J),K=1,3),TEMP(I,J),J=1,NJ)    OUTP1380
681 FORMAT(4(3X,1PE10.2,4H LE ,3A1,3H LT,E10.2))                     OUTP1390
683 CONTINUE                                                         OUTP1400
600 CONTINUE                                                         OUTP1410
C                                                                    OUTP1420
100 RETURN                                                           OUTP1430
    END                                                              OUTP1440
```

```
      FUNCTION PURPLE(X,Y)                                                  PURP0010
C  FUNCTION TABLES, PARAMETERS AND VALUE STORAGE                            PURP0020
      COMMON /PLOTCM/ NRNG,DRNG,NPH,DPH,DPH2,IVO,IVA,IVF,IPOSOP(2),         PURP0030
     * NSETS,IPORD,                                                         PURP0040
     * TTLORD(12),TTLFEG(12),NPLOTS,NX,NY,XMIN,YMIN,DX,DY,NZ,CH(55),        PURP0050
     * RECT(81,41),ITF(20),ISF(20),ITYF(20),IVPF(20),NF(20),F(14,20),       PURP0060
     * ITTLF(12,20),KTF(20),OZMIN(20),OZMAX(20),XTB(14,10),YTB(14,10),      PURP0070
     * C(7,10),NTB(10),KTB(10),OUT(20,36,8),ITTLP(12,8),KTPP(8),OLP(8),     PURP0080
     * OHP(8),ISP(8),ITP(8),PZMIN(8),PZMAX(8),ITYP(8),ITYS(8),IVPP(8)       PURP0090
C                                                                           PURP0100
C  COMMON BLOCK PASSED FROM 'SOJAC'                                         PURP0110
      COMMON /CSOJAC/ GX,GY,IXP,IYP,IX,IY                                   PURP0120
C                                                                           PURP0130
C     SUPPLIES VALUES OF RECT TO 'SOJAC' (PRINTER CONTOUR PLOTTER)          PURP0140
C                                                                           PURP0150
      XX = (X-XMIN)/GY+1.                                                   PURP0160
      IXX = XX                                                              PURP0170
      XX = 1.-AMOD(XX,1.)                                                   PURP0180
      IXX = MIN0(NY-1,IXX)                                                  PURP0190
      IXY = NY-IXX                                                          PURP0200
      IF(RECT(IY,IXX).LT.1.E+38 .AND. RECT(IY,IXY+1).LT.1.E+38) GO TO 1     PURP0210
      PURPLE = 1.E+38                                                       PURP0220
      RETURN                                                                PURP0230
    1 PURPLE = RECT(IY,IXX)*(1.-XX)+RECT(IY,IXX+1)*XY                       PURP0240
      RETURN                                                                PURP0250
      END                                                                   PURP0260
```

```
      FUNCTION POLCAR(F,MR,NR,NT,DR,DT,DRM,DTM,X,Y)
      DIMENSION P(MR,1)
C
C     GIVEN TABULAR FUNCTION P DEFINED ON A POLAR GRID OF NR RADII AND
C     NT AZIMUTHS,, RADIUS I = DR*(I-1)+DRM,  AZIMUTH J = DT*(J-1)+DTM,
C     AND CARTESIAN COORDINATES (X,Y), RETURN VALUE OF P AT (X,Y).
C     A BILINEAR (IN R,T) INTERPOLATION IS USED.
C     POLCAR IS SET 'UNDEFINED' (=1.E+38) IF ANY OF THE FOUR DATA POINTS
C     SURROUNDING (X,Y) ARE UNDEFINED, OR IF (X,Y) LIES WITHIN RADIUS
C     'DRM' OR BEYOND RADIUS 'DRM*NR'
C
      IF (X.EQ.0. .AND. Y.EQ.0.) GO TO 5
      RS = SQRT(X*X+Y*Y)
      TS = ATAN2(Y,X)
      I = (RS-DRM)/DR+1.
      J = (TS-DTM)/DT+1.
      IF(I.LT.1 .OR. I.GT.NR) GO TO 5
      J = MAX0(1,MIN0(NT-1,J))
      TH = (TS-DT*FLOAT(J-1)-DTM)/DT
      I = MAX0(1,MIN0(NR-1,I))
      IF(P(I,J).GE.1.E+38 .OR. P(I,J+1).GE.1.E+38 .OR.
     * P(I+1,J).GE.1.E+38 .OR. P(I+1,J+1).GE.1.E+38) GO TO 5
      RH = (RS-DR*FLOAT(I-1)-DRM)/DR
      POLCAR = (P(I,J)*(1.-TH) + P(I,J+1)*TH) * (1.-RH)
     *        + (P(I+1,J)*(1.-TH) + P(I+1,J+1)*TH) * RH
      RETURN
    5 POLCAR = 1.E+38
      RETURN
      END
```

```
C     **********                                                        CCAR0010
C     * CCARD  *          JOHN HOUSE      IITRI       SEPT 1970          CCAR0020
C     **********                                                        CCAR0030
C                                                                       CCAR0040
      LOGICAL FUNCTION CCARD(S1,C1,L1,IACT,W)                           CCAR0050
C                                                                       CCAR0060
C     INTERPRET CONTROL CARD FIELDS IDENTIFIED BY SYNTAX.               CCAR0070
C                                                                       CCAR0080
C     S1        TEXT STRING.                                            CCAR0090
C     C1        POINTER TO START OF CURRENT FIELD                       CCAR0100
C     L1        LENGTH OF CURRENT FIELD                                 CCAR0110
C     IACT      ACTION CODE RETRIEVED BY SYNTAX FROM SYNTAX RULES       CCAR0120
C     W         GENERAL ARRAY FOR CONTROL CARDS..                       CCAR0130
C               W(1)--W(10).. FLAGS AND COUNTERS FOR USE BY AND COMMUNI-CCAR0140
C                             CATION WITH SYNTAX                        CCAR0150
C               W(11).. NUMBER OF CONTROL CARD ENTRIES TIMES 7 PLUS 20. CCAR0160
C               W(12)--W(20).. HOLD OUTPUT VALUES FOR USE BY PROGRAM    CCAR0170
C                              CALLING SYNTAX                           CCAR0180
C               W(12).. CONTROL CARD TYPE CODE                          CCAR0190
C               W(13).. NON-SYNTAX ERROR COLUMN (=0 IF NONE)            CCAR0200
C               W(14),W(15).. POINT TO START AND END OF FIELD ORDINAL   CCAR0210
C                             LIST (NOT USED IF W(14).GT.W(15))         CCAR0220
C               W(16),W(17).. POINT TO START AND END OF POSITIONAL FIELD CCAR0230
C                             LIST (NOT USED IF W(16).GT.W(17))         CCAR0240
C                                                                       CCAR0250
C     UPON RETURN, CCARD=.TRUE. IF FIELD IS OK, OR CCARD=.FALSE. IF NOTCCAR0260
C                                                                       CCAR0270
C     CONTROL CARD TYPE 1 IS RESERVED FOR C.CDS. WHICH NEVER REQUIRE    CCAR0280
C     FURTHER USER ACTION.  (I.E., CARD MERELY SETS FLAGS AND VALUES)   CCAR0290
C     FIELD TYPE CODE 9 IS RESERVED FOR THE 'FIELD DEFAULT VALUE        CCAR0300
C     REDEFINITION' FLAG.  THE REMAINDER OF A C.CD. ON WHICH THIS FLAG  CCAR0310
C     APPEARS CONSISTS OF FIELD DEF. VAL. REDEFNS.  (NOTE.. FLAG DEFAULTCCAR0320
C     SETTINGS ARE ENTERED IN THE LOGICAL FORM (NAME=T, NAME=.FALSE.,   CCAR0330
C     ETC.)  DO NOT SIMPLY ENTER 'NAME' OR 'NO'+'NAME')                 CCAR0340
C                                                                       CCAR0350
      INTEGER S1(1)                                                     CCAR0360
      INTEGER C1,W(1),NULLV(5),INULL(3)                                 CCAR0370
      LOGICAL GRIPE,POSIT,EXIST,LNULL,DEFINE                            CCAR0380
      REAL FNULL                                                        CCAR0390
C     KV AND RV ARE EQUIVALENT NAMES FOR THE 1ST WORD OF KVA.           CCAR0400
```

```
C     KV IS ALWAYS USED FOR LOGICAL AND INTEGER VALUES.                CCAR0410
C     RV IS ALWAYS USED FOR REAL VALUES.                               CCAR0420
      INTEGER KVA(2)                                                   CCAR0430
      EQUIVALENCE (KVA(1),KV,RV)                                       CCAR0440
      DATA IBL/1H /,INULL,LNULL,FNULL/0,0,1H ,.FALSE.,0.0/             CCAR0450
C***  VALUE OF NCHV IS MACHINE DEPENDENT -- MAX. NO CHAR. IN NAME OR VALCCAR0460
      DATA NCHV/12/                                                    CCAR0470
C***  KNL IS MACH. DEP. -- NUMBER OF 'WORDS' ALLOWED FOR A NAME OF VALUECCAR0480
      DATA KNL/2/                                                      CCAR0490
      EQUIVALENCE (NULLV(1),INULL(1)),(NULLV(4),INULL),(NULLV(5),FNULL),CCAR0500
     * (IEXIST,EXIST)                                                  CCAR0510
C                                                                      CCAR0520
      LMAX = MIN0(L1,NCHV)                                             CCAR0530
      GO TO (100,200,300,400,500,600,700,800,900,1000),IACT            CCAR0540
C                                                                      CCAR0550
C  CONTROL CARD NAME                                                   CCAR0560
C                                                                      CCAR0570
  100 N = W(11)                                                        CCAR0580
      W(12) = 0                                                        CCAR0590
      W(13) = 0                                                        CCAR0600
      DO 101 KK=21,N,7                                                 CCAR0610
      I = W(    KK+1)                                                  CCAR0620
  102 IF(I.LE.0) GO TO 101                                             CCAR0630
      II = I+KNL                                                       CCAR0640
      J = KOML(W(I),1,IABS(W(II)),S1,C1,LMAX)                          CCAR0650
      IF (J.EQ.0) GO TO 110                                            CCAR0660
      IF (J.NE.-1) GO TO 104                                           CCAR0670
  103 IF(W(II).LT.0) GO TO 110                                         CCAR0680
  104 I = W(II+1)                                                      CCAR0690
      GO TO 102                                                        CCAR0700
  101 CONTINUE                                                         CCAR0710
      GO TO 9                                                          CCAR0720
C  CONTROL CARD NAME FOUND                                             CCAR0730
  110 W(12)= W(KK)                                                     CCAR0740
      KPOS1 = W(KK+2)                                                  CCAR0750
      KPOS = KPOS1                                                     CCAR0760
      W(16)= KPOS1                                                     CCAR0770
      KPOS2 = W(KK+3)-1                                                CCAR0780
      KEY1 = W(KK+3)                                                   CCAR0790
      KEY2 = W(KK+4)                                                   CCAR0800
```

```
      KOR1 = W(KK+5)                                                    CCAR0810
      KOR = KOR1                                                        CCAR0820
      W(14) = KOR1                                                      CCAR0830
      KOR2 = W(KK+6)                                                    CCAR0840
C   SET DEFAULT FIELDS                                                  CCAR0850
       IF(KPOS1.GT.KEY2) GO TO 120                                      CCAR0860
      DO 115 K=KPOS1,KEY2                                               CCAR0870
      I = W(K)                                                          CCAR0880
      W(I+2) = 0                                                        CCAR0890
      IF(W(I).GE.81000) GO TO 115                                       CCAR0900
      W(I+3) = W(I)/1000                                                CCAR0910
      II = I+KNL+4                                                      CCAR0920
      IF(MOD(W(I),10)-4) 113,116,114                                    CCAR0930
C***  LOADING OF DEFAULT VALUE  MAY REQUIRE A DO LOOP                    CCAR0940
  113 W(I+4) = W(II)                                                    CCAR0950
      W(I+5) = W(II+1)                                                  CCAR0960
      GO TO 115                                                         CCAR0970
  114 W(I+4) = W(II)                                                    CCAR0980
      GO TO 115                                                         CCAR0990
  116 L = MIN0(W(I+4),W(II))                                            CCAR1000
      K1 = W(I+5)                                                       CCAR1010
      K2 = W(II+1)                                                      CCAR1020
      CALL MOVE(W(K1),1,L,W(K2),1)                                      CCAR1030
      J = W(I+4)-L                                                      CCAR1040
      IF(J.LE.0) GO TO 115                                              CCAR1050
      CALL MOVE(W(K1),L+1,1,1H ,1)                                      CCAR1060
      CALL MOVE(W(K1),L+2,J-1,W(K1),L+1)                                CCAR1070
  115 CONTINUE                                                          CCAR1080
  120 GRIPE = .FALSE.                                                   CCAR1090
      DEFINE = .FALSE.                                                  CCAR1100
      GO TO 1                                                           CCAR1110
C                                                                       CCAR1120
C   POSITIONAL VALUE OR FIELD NAME                                      CCAR1130
C                                                                       CCAR1140
  200 KS = C1                                                           CCAR1150
      LS = LMAX                                                         CCAR1160
      LS1 = L1                                                          CCAR1170
      GO TO 1                                                           CCAR1180
C                                                                       CCAR1190
C   PRIOR WORD WAS A FIELD NAME (FOLLOWED BY A VALUE)                   CCAR1200
```

```
C                                                                        CCAR1210
  300 POSIT = .FALSE.                                                    CCAR1220
C  ALL REQ. POSIT. FIELDS MUST BE SUPPLIED PRIOR TO 1ST KEY TYPE FIELD   CCAR1230
      IF(KPOS.GT.KPOS2) GO TO 510                                        CCAR1240
      KW = W(KPOS)                                                       CCAR1250
      IF(W(KW).GE.62000) GO TO 5                                         CCAR1260
      GO TO 510                                                          CCAR1270
C                                                                        CCAR1280
C  FIELD VALUE.  INTERPRET AS INTEGER, ALPHA, LOGICAL, OR REAL.          CCAR1290
C                                                                        CCAR1300
  400 KS = C1                                                            CCAR1310
      LS = LMAX                                                          CCAR1320
      LS1 = L1                                                           CCAR1330
  410 K2 = K2+1                                                          CCAR1340
      IF(K2.LE.0) GO TO 1                                                CCAR1350
      KV = 0                                                             CCAR1360
      L = 0                                                              CCAR1370
      GO TO (440,440,420,430,440,450,460,470),K2                         CCAR1380
C  INTEGER                                                               CCAR1390
  420 IF (INTI(S1,KS,LS1,0,KV).NE.1) GO TO 495                           CCAR1400
      GO TO 405                                                          CCAR1410
C  ALPHAMERIC                                                            CCAR1420
C***  BLANK OUT ALL WORDS OF KVA BEFORE MOVING TEXT                       CCAR1430
  430 KV = IBL                                                           CCAR1440
      KVA(2) = IBL                                                       CCAR1450
      CALL MOVE(KV,1,LS,S1,KS)                                           CCAR1460
      L = LS                                                             CCAR1470
      GO TO 405                                                          CCAR1480
C  LOGICAL                                                               CCAR1490
  440 IF (ILGI(S1,KS,LS1,0,KV).NE.1) GO TO 405                           CCAR1500
      GO TO 405                                                          CCAR1510
C  REAL                                                                  CCAR1520
  450 IF (IRFI(S1,KS,LS1,0,0,RV).NE.1) GO TO 495                         CCAR1530
      GO TO 405                                                          CCAR1540
C  LONG-ALPHAMERIC                                                       CCAR1550
  460 IF(DEFINE) GO TO 465                                               CCAR1560
      LS1 = MIN0(MAP,LS1)                                                CCAR1570
      CALL MOVE(W(JAP),1,LS1,S1,KS)                                      CCAR1580
      W(KW+2) = 1                                                        CCAR1590
      W(KW+3) = LS1                                                      CCAR1600
```

```
      GO TO 1                                                           CCAR1610
  465 IF(W(K ).GE.81000) [illegible]                                    CCAR1620
      LS1 = [illegible](KAP,LS1)                                        CCAR1630
      CALL [illegible](W(JAP),[illegible],S1, S)                        CCAR1640
      W(KW) = LS1*1[illegible](K ),1[illegible])                        CCAR1650
      J = KAP-LS1                                                       CCAR1660
      IF(J.LE.0) GO TO 1                                                CCAR1670
      CALL MOVE(W(JAP),[illegible])                                     CCAR1680
      CALL MOVE(W(JAP),[illegible])                                     CCAR1690
      GO TO 1                                                           CCAR1700
C  SYMBOLIC -- SEARCH DICTIONARY FOR MATCH WITH SYMBOL                  CCAR1710
  470 I = W(KW+5)                                                       CCAR1720
      L = 0                                                             CCAR1730
      KV = 0                                                            CCAR1740
  471 IF(I.LE.0) GO TO 405                                              CCAR1750
      II = I+K*L                                                        CCAR1760
      KV = KV+1                                                         CCAR1770
      J = KOMPL(W(I),1,IABS(W(II)),S1,KC,LS)                            CCAR1780
      IF (J.EQ.0) GO TO 405                                             CCAR1790
      IF (J.NE.-1) GO TO 472                                            CCAR1800
  473 IF(W(II).LT.0) GO TO 405                                          CCAR1810
  472 I = W(II+1)                                                       CCAR1820
      GO TO 471                                                         CCAR1830
C  CONCLUSION                                                           CCAR1840
  405 IF(DEFINE) GO TO 407                                              CCAR1850
      W(KW+2) = 1                                                       CCAR1860
      W(KW+3) = L                                                       CCAR1870
C   A DO LOOP MAY BE REQUIRED TO MOVE ALL OF KVA                        CCAR1880
      W(KW+4) = KV                                                      CCAR1890
      IF(K2.GE.7) GO TO 1                                               CCAR1900
      W(KW+5) = KVA(2)                                                  CCAR1910
      GO TO 1                                                           CCAR1920
C   A DO LOOP MAY BE REQUIRED TO MOVE ALL OF KVA                        CCAR1930
  407 IF(W(KW).GE.81000) GO TO 5                                        CCAR1940
      II = KW+KNL+4                                                     CCAR1950
      W(II) = KV                                                        CCAR1960
      IF(K2.LT.7) W(II+1)=KVA(2)                                        CCAR1970
      W(KW) = L*1000+MOD(W(KW),1000)                                    CCAR1980
      GO TO 1                                                           CCAR1990
C  VALUE ERROR                                                          CCAR2000
```

Reproduced from best available copy.

```
  495 W(KW+2) =-1                                                       CCAR2010
      IF(DEFINE) GO TO 5                                                CCAR2020
      W(KW+3) = 0                                                       CCAR2030
      W(KW+4) = 0                                                       CCAR2040
      IF(K2.GE.7) GO TO 5                                               CCAR2050
      W(KW+5) = 0                                                       CCAR2060
      GO TO 5                                                           CCAR2070
C                                                                       CCAR2080
C  PRIOR WORD WAS EITHER A FIELD NAME (W/O A VALUE) OR A POSIT. VALUE   CCAR2090
C                                                                       CCAR2100
  500 POSIT = .TRUE.                                                    CCAR2110
      IF(DEFINE) GO TO 5                                                CCAR2120
C  IF ANY REQ. POSIT. FIELDS REMAIN UNSATISFIED, DO NOT SRC FOR KEYWORD CCAR2130
      IF(KPOS.GT.KPOS2) GO TO 515                                       CCAR2140
      KW = W(KPOS)                                                      CCAR2150
      IF(W(KW).GE.82000) GO TO 550                                      CCAR2160
C                                                                       CCAR2170
C  SEARCH FOR POSSIBLE FIELD NAME                                       CCAR2180
C                                                                       CCAR2190
  510 IF(.NOT.DEFINE) GO TO 515                                         CCAR2200
      IF (INTI(S1,KS,LS,0,W),NE,1) GO TO 515                            CCAR2210
      IF(K.LE.0) GO TO 5                                                CCAR2220
      K = KPOS1-1+K                                                     CCAR2230
      IF(K.GT.KPOS2) GO TO 5                                            CCAR2240
      KW = W(K)                                                         CCAR2250
      K2 = MOD(W(KW),10)                                                CCAR2260
      IF(KOR.GT.KOR2) GO TO 1                                           CCAR2270
      W(KOR) = KW                                                       CCAR2280
      KOR = KOR+1                                                       CCAR2290
      GO TO 1                                                           CCAR2300
  515 DO 511 KEY=KEY1,KEY2                                              CCAR2310
      KW = W(KEY)                                                       CCAR2320
      IF(W(KW+2).NE.0) GO TO 511                                        CCAR2330
      K2 = MOD(W(KW),10)                                                CCAR2340
      K = W(KW+1)                                                       CCAR2350
C  IF K=V TYPE FIELD ENCOUNTERED, DO NOT SEARCH K OR V TYPE FIELD ENTR. CCAR2360
C  IF K OR V TYPE FIELD ENCOUNTERED, DO NOT SRC K=V TYPE ENTRIES        CCAR2370
      EXIST = K.LE.0 .OR. K2.LE.1 .OR. K2.GE.9                          CCAR2380
      IF(POSIT .AND. .NOT.EXIST .OR. .NOT.POSIT .AND. EXIST) GO TO 511  CCAR2390
  512 IF(K.LE.0) GO TO     511                                          CCAR2400
```

```
      II = [illegible]+K*L                                              CCAR2410
      K1 = IABS([illegible](II))                                        CCAR2420
      I = K[illegible]L[illegible] (K),[illegible], S,LS)               CCAR2430
      IF (IABS(I).GT.1) GO TO 513                                       CCAR2440
      IF (I) 5[illegible],520,5[illegible]                              CCAR2450
  513 IF(K2.[illegible]E.1) GO TO 519                                   CCAR2460
      IF(LS.[illegible]E.2) GO TO 519                                   CCAR2470
      IF(K[illegible] (2[illegible],1,2,S[illegible],K5).[illegible]) GO TO [illegible] CCAR2480
      J = KC[illegible]L(W(K),1,K[illegible],S[illegible], S+2,L[illegible]) CCAR2490
      IF (I.EQ.0) GO TO 52[illegible]                                   CCAR2500
      IF (I.NE.-1) GO TO 519                                            CCAR2510
  514 IF(W(II).LT.0) GO TO 521                                          CCAR2520
      GO TO 519                                                         CCAR2530
  518 IF(W(II).LT.0) GO TO 52[illegible]                                CCAR2540
  519 K = W(II+1)                                                       CCAR2550
      GO TO 512                                                         CCAR2560
  511 CONTINUE                                                          CCAR2570
C  NO MATCH FOUND -- OK IF THIS IS POSITIONAL                           CCAR2580
      IF(KPOS.GT.KPOS2 .OR. .NOT.POSIT) GO TO 5[illegible]              CCAR2590
  550 KW = [illegible](KPOS)                                            CCAR2600
      KPOS = KPOS+1                                                     CCAR2610
      K2 = MOD(W(KW),10)                                                CCAR2620
      IF(KOR.GT.KOR2) GO TO 551                                         CCAR2630
      W(KOR) =    KW                                                    CCAR2640
      KOR = KOR+1                                                       CCAR2650
  551 IF(POSIT) GO TO 560                                               CCAR2660
      K2 = MOD(W(KW),10)                                                CCAR2670
      GO TO 523                                                         CCAR2680
C  PREPARE FIELD (IF LONG-ALPHA)                                        CCAR2690
  560 IF(MOD(W(KW),10).NE.6) GO TO 410                                  CCAR2700
      ASSIGN 410 TO KG[illegible]                                       CCAR2710
      GO TO 546                                                         CCAR2720
C  FIELD NAME (OR [illegible]+NAME) FOUND                               CCAR2730
  520 EXIST = .TRUE.                                                    CCAR2740
      GO TO 522                                                         CCAR2750
  521 EXIST = .FALSE.                                                   CCAR2760
  522 IF(DEFINE) GO TO 523                                              CCAR2770
      IF(K2.EQ.9) GO TO 540                                             CCAR2780
  523 IF(KOR.GT.KOR2) GO TO 530                                         CCAR2790
      W(KOR) = KW                                                       CCAR2800
```

```
      KOP = KOP+1                                                       CCAR2810
C   SET EXISTANCE OR NULL VALUES                                        CCAR2820
  530 IF(K2.GT.1) GO TO 535                                             CCAR2830
      W(KW+2) = 1                                                       CCAR2840
      W(KW+4) = IEXIST                                                  CCAR2850
      GO TO 1                                                           CCAR2860
  535 IF(K2-6) 536,545,555                                              CCAR2870
  536 W(KW+4) = NULLV(K2)                                               CCAR2880
      IF(K2.EQ.3) W(KW+5)=NULLV(K2)                                     CCAR2890
      W(KW+3) = 0                                                       CCAR2900
      JAP = KW+4                                                        CCAR2910
      LAP = NCHW                                                        CCAR2920
      MAP = NCHW                                                        CCAR2930
      KAP = 1                                                           CCAR2940
      GO TO 1                                                           CCAR2950
C   FIELD IS LONG-ALPHABETIC                                            CCAR2960
  545 ASSIGN 1 TO KGO                                                   CCAR2970
  546 II = KW+4                                                         CCAR2980
      IF (DEFINE) II=II+KAL                                             CCAR2990
      JAP = W(II+1)                                                     CCAR3000
      LAP = W(II)                                                       CCAR3010
      MAP = LAP                                                         CCAR3020
      KAP = 1                                                           CCAR3030
      IF(LAP.LE.0) GO TO 549                                            CCAR3040
      CALL MOVE(W(JAP),1,1,1H ,1)                                       CCAR3050
      CALL MOVE(W(JAP),2,LAP-1,W(JAP),1)                                CCAR3060
      IF (DEFINE) GO TO 547                                             CCAR3070
      W(KW+3) = 0                                                       CCAR3080
      GO TO 549                                                         CCAR3090
  547 W(KW) = MOD(W(KW),1000)                                           CCAR3100
  549 GO TO KGO                                                         CCAR3110
C   SYMBOLIC -- COLLECT ARGUMENT FOR A SEARCH                           CCAR3120
  555 W(KW+4) = 0                                                       CCAR3130
      GO TO 1                                                           CCAR3140
C   IF A FIELD OF TYPE 9 IS ENCOUNTERED, THEN REST OF CARD SETS DEFAULTS CCAR3150
  540 DEFINE = .TRUE.                                                   CCAR3160
      W(12) = 1                                                         CCAR3170
      GO TO 1                                                           CCAR3180
C                                                                       CCAR3190
C   END OF CONTROL CARD -- CHECK FOR MISSING REQUIRED FIELDS            CCAR3200
```

```
C
  600 IF(KPOS1.GT.KEY2) GO TO 6
      DO 610 K=KPOS1,KEY2
      I = W(K)
      IF(W(I).LT.82000) GO TO 610
      IF(W(I+2).LE.0) GO TO 630
  610 CONTINUE
  620 IF(GRIPE) GO TO 640
      CCARD = .TRUE.
      GO TO 640
  630 IF (GRIPE) GO TO 635
      IGRIPE = C1
  635 W(13) = IGRIPE
      CCARD = .FALSE.
  640 W(15) = KOP-1
      W(17) = KPOS-1
      RETURN
C
C  ADD TO I-TYPE STRING (PROHIBITED FOR NON-ALPHAMERIC FIELDS)
C
  700 IF(K2.NE.3 .AND. K2.NE.6) GO TO 6
      CALL CAT(W(JAP),MAP,LAP,81,C1,1)
      IF(DEFINE) GO TO 705
      W(KW+2) = 2
      W(KW+3) = MAP-LAP
      GO TO 1
  705 W(KW) = (MAP-LAP)*1000+MOD(W(KW-1),1000)
      GO TO 1
C
C  START OF POSITIONAL I-TYPE ALPHA VALUE (RESERVE POSIT, FIELD ENTRY)
C
  800 IF(DEFINE) GO TO 6
      POSIT = .FALSE.
      IF(KPOS.GT.KPOS2) GO TO 6
      GO TO 550
C
C  RETURNS
C
  900 CONTINUE
 1000 CONTINUE
```

```
9 CCARD = .FALSE.                                                    CCAR3610
  RETURN                                                             CCAR3620
6 IF(GRIPE) GO TO 1                                                  CCAR3630
  IGRIPE = C1+L1                                                     CCAR3640
  GO TO 7                                                            CCAR3650
5 IF(GRIPE) GO TO 1                                                  CCAR3660
  IGRIPE = KS+LS1                                                    CCAR3670
7 GRIPE = .TRUE.                                                     CCAR3680
1 CCARD = .TRUE.                                                     CCAR3690
  RETURN                                                             CCAR3700
  END                                                                CCAR3710
```

```
C     **********
C     *  CAT   *           JOHN ROUSE       IITRI        SEPT 197?
C     **********
C
      SUBROUTINE CAT (S1,C1,L1,S2,C2,L2)
      INTEGER S1(1),C1,L1,S2(1),C2,L2
C
C     CONCATENATE L CHAR., STARTING WITH CHAR. C2 OF STRING S2, ONTO
C     STRING S1, WHICH HAS ROOM FOR L1 MORE CHAR., STARTING AT CHAR.
C     POS. C1.   L=MIN(L1,L2)
C     UPON RETURN, C1 IS INCREASED BY L, AND L1 IS DECREASED BY L.
C     BOTH STRINGS S1 AND S2 ARE PACKED (I.E., 6 CHAR. PER WORD)
C
      L = MINO(L1,L2)
      IF(L.LE.0) RETURN
      CALL RCHXTK(S1,C1)
      CALL RCHXTK(S2,C2)
      DO 1 I=1,L
    1 CALL SCHXTK(S1,K1,IC=XTK(S2,K2))
      C1 = C1+L
      L1 = L1-L
      RETURN
      END
```

```
C     **********
C     *  IFXO  *          JOHN ROUSE      IITRI        MAY 1971
C     **********
C
      FUNCTION IFXO(S,C,L,N,W)
      INTEGER S(1),U
      INTEGER C,X,W,U
      LOGICAL OS
C***  W.P. INTEGER VALUE OF CHAR. ZERO
      DATA U/060/
C
C     CONVERT THE ABSOLUTE VALUE OF INTEGER 'N' TO CHAR. FORM AND PLACE
C     IT IN THE OUTPUT FIELD.
C     IF W LE 0, THE OUTPUT FIELD IS THE 'L' CHAR. OF STRING S, STARTING
C              AT CHAR. 'C'.  RESULT IS RIGHT-JUSTIFIED W/BLANK FILL.
C     IF W GT 0, THE OUTPUT FIELD IS THE UP TO W CHAR. (OR UP TO 'L'
C              CHAR. IF L LT W) OF STRING S, STARTING AT CHAR. 'C'.
C              RESULT TAKES ONLY AS MANY CHAR. AS REQUIRED.
C     THE OUTPUT FIELD IS COMPLETELY FILLED.  LEADING ZEROES ARE
C     SUPPLIED IF NECESSARY.  IF FIELD IS TOO NARROW, LEADING DIGITS
C     ARE TRUNCATED.
C     NOTE.. IF W GT 0, C AND L MUST BOTH BE VARIABLES, NOT CONSTANTS.
C     UPON RETURN, IFXO IS SET TO..
C         =-1  IF OUTPUT FIELD IS TOO NARROW FOR THE CONVERTED VALUE.
C              C AND L ARE RETURNED UNCHANGED.
C              IF W LE 0, THE FIELD IS FILLED WITH ASTERISKS.
C         =+1  IF OUTPUT FIELD IS WIDE ENOUGH.  IF W GT 0, C IS SET TO
C              REFER TO THE 1ST CHAR. AFTER THE OUTPUT FIELD, AND
C              L IS A COUNT OF THE REMAINING CHARS IN S, AFTER THE FLD.
C
      OS = W.GT.0
      LL = L
      IF (OS) LL=MIN0(L,W)
      IF (LL.LE.0) GO TO 30
      CALL RCHXTK(U,C+LL-1)
      M = IABS(N)
      DO 1 I=1,LL
      J=M/10
      X=M-J*10
      CALL PCHXTK(S,U,X+U)
```

```
1  M=0                                                IFX00410
   IF(.NOT.PS) RETURN                                 IFX00420
   C = C+LL                                           IFX00430
   L = L-LL                                           IFX00440
   RETURN                                             IFX00450
30 IFXO = -1                                          IFX00460
   RETURN                                             IFX00470
   END                                                IFX00480
```

```
C      **********
C      *  ILGI  *          JOHN LOUGE      IITRI       MAY 1971
C      **********
C
       FUNCTION ILGI(S,C,L,W,G)
       INTEGER S(1),C,W,Z
       LOGICAL GS,G
C***   M,D, CHAR, F, T
       DATA IF,IT/C13,C31/
C
C      CONVERT THE LL CHAR, OF STRING 'S', STARTING AT CHAR. 'C', TO
C      THE LOGICAL VALUE 'G'.
C      IF W LE 0, LL=L.  IF W GT 0, LL=W, OR LL=L IF L LT W.
C      BLANKS ARE IGNORED.  AN ALL BLANK OR A NULL STRING IS OK.
C      G IS SET FALSE UNLESS A 'T' IS FOUND BEFORE THE 1ST 'F' (IF ANY)
C      NOTE.. IF W GT 0, C AND L MUST BOTH BE VARIABLES, NOT CONSTANTS.
C      UPON RETURN, ILGI IS SET TO 1.  NO ERROR CONDITIONS ARE CHECKED.
C
       GS = W.GT.0
       LL = L
       IF (GS) LL=MIN0(L,W)
       ILGI = 1
       G = .FALSE.
       IF(LL.LE.0) RETURN
       CALL RCHXTK(Z,C)
       DO 6 I=1,LL
       J = ICHXTK(S,Z)
       IF(J.EQ.IT) GO TO 7
       IF(J.EQ.IF) GO TO 9
     6 CONTINUE
       I = LL
     9 GO TO 10
     7 G = .TRUE.
    10 IF(.NOT.GS) RETURN
       L = L-I
       C = C+I
       RETURN
       END
```

```
C     **********
C     *  INTI  *          JOHN HOUSE       IITRI        MAY 1971
C     **********
C
      FUNCTION INTI(S,C,L,W)
      INTEGER S(1),C,W
      LOGICAL QS
C***  M, D, CHAR, BLANK,PLUS,
      DATA IB,IP,IM,ID/
C
C     CONVERT THE LL CHAR. OF STRING 'S', STARTING AT CHAR. 'C', TO
C     A SIGNED INTEGER 'I'.
C     IF W LE 0, LL=L.  IF W GT 0, LL=W, OR LL=L IF L LT W.
C     BLANKS ARE IGNORED.  AN ALL BLANK OR A NULL STRING IS OK.
C     NOTE.. IF W GT 0, C AND L MUST BOTH BE VARIABLES, NOT CONSTANTS.
C     UPON RETURN, INTI IS SET TO..
C          =-1  IF NO LEGAL CHAR. IS FOUND (EXCEPT POSSIBLY BLANKS)
C          = 0  IF AN ILLEGAL CHAR. IS FOUND, BUT AFTER AT LEAST 1 LEGAL.
C          IF ONE OF THESE CODES IS RETURNED, AND IF W GT 0,
C          THEN THE ILLEGAL CHAR. IS CHAR. C OF STRING S, AND
C          L IS A COUNT OF REMAINING CHARS IN S, INCL. THE ILL. CHAR.
C          =+1  IF THERE ARE NO ILLEGAL CHARS IN THE INPUT FIELD.
C               IF W GT 0, C IS SET TO REFER TO THE 1ST CHAR AFTER THE
C               INPUT FIELD, AND L IS A COUNT OF THE REMAINING CHAR. IN S
C
      QS = W.GT.0
      LL = L
      IF (QS) LL=MIN0(L,W)
C  CONVERT FIELD
    1 N = 0
      NS = 0
      INTI = 1
      IF (LL.LE.0) RETURN
      CALL RCHXTK(U,C)
      K2 = C+LL-1
      DO 10 K=C,K2
      I = ICHXTK(S,U)
      IF(I.EQ.IB) GO TO 10
      IF (NS.NE.0) GO TO 18
      IF(I.EQ.IM) GO TO 15
```

```
      IF(I.EQ.IP .OR. I.EQ.IPE) GO TO 16                             INTI0410
   18 IF(I.LT.I00 .OR. I.GT.I09) GO TO 20                             INTI0420
      N = N*10+I-I00                                                  INTI0430
      IF (NS.EQ.0) GO TO 16                                           INTI0440
      GO TO 10                                                        INTI0450
   15 NS = -1                                                         INTI0460
      GO TO 10                                                        INTI0470
   16 NS = +1                                                         INTI0480
   10 CONTINUE                                                        INTI0490
      K = C+LL                                                        INTI0500
      GO TO 25                                                        INTI0510
   20 IF (NS.EQ.0) GO TO 30                                           INTI0520
      INTI = 0                                                        INTI0530
   25 IF(NS.LT.0) N=-N                                                INTI0540
      GO TO 35                                                        INTI0550
   30 INTI = -1                                                       INTI0560
   35 IF(.NOT.SS) RETURN                                              INTI0570
      L = L-K+C                                                       INTI0580
      C = K                                                           INTI0590
      RETURN                                                          INTI0600
      END                                                             INTI0610
```

```
C     **********                                                     IREI0010
C     * IREI *        JOHN ROUSE      IITRI       MAY 1971           IREI0020
C     **********                                                     IREI0030
C                                                                    IREI0040
      FUNCTION IREI(S,C,L,W,D,V)                                     IREI0050
      DOUBLE PRECISION EV,VV,BIG,SMALL,T(10),TT(6)                   IREI0060
      INTEGER S(1),C,W,D,SC(2),D                                     IREI0070
      LOGICAL GS,GY                                                  IREI0080
      EQUIVALENCE(BIG,BI(1)),(SMALL,SM(1))                           IREI0090
C***  W.D. CHAR. BLANK, ZERO, NINE, PLUS, 'E', MINUS,  MXI=MAX INT DGTS IREI0100
      DATA IB,IZ,I9,IP,IE,IM/C05,060,071,1H.,1HE,041/,MXI/10/        IREI0110
C***  W.D. LARGEST AND SMALLEST REAL VALUES                           IREI0120
      DATA BI,SM/O2177777777777,O777777777777,O1600000000000,0/      IREI0130
      DATA T/1.D0,1.D1,1.D2,1.D3,1.D4,1.D5,1.D6,1.D7,1.D8,1.D9/      IREI0140
      DATA TT/1.D-40,1.D-30,1.D-20,1.D-10,1.D+00,1.D+10,1.D+20,1.D+30/ IREI0150
C                                                                    IREI0160
C     CONVERT THE LL CHAR. OF STRING 'S', STARTING AT CHAR. 'C', TO  IREI0170
C     A REAL VALUE 'V'.                                              IREI0180
C     IF W LE 0, LL=L.  IF W GT 0, LL=W, OR LL=L IF L LT W.          IREI0190
C     BLANKS ARE IGNORED.  AN ALL BLANK OR A NULL STRING IS OK.      IREI0200
C     NO SIGN OR DECIMAL POINT IS REQUIRED.  HOWEVER, IF THERE IS NO IREI0210
C     DECIMAL POINT, ONE IS ASSUMED TO EXIST JUST TO THE LEFT OF THE IREI0220
C     'D'TH DIGIT (COUNTING FROM THE RIGHT.)                         IREI0230
C     AN EXPONENT FIELD IS OPTIONAL.  IF PRESENT, IT MUST BE IN ONE OF IREI0240
C     THE FOLLOWING FORMS,,  +N, -N, E+N, E-N, EN, WHERE 'N' IS ONE  IREI0250
C     OR MORE DIGITS.                                                IREI0260
C     NOTE.. IF W GT 0, C AND L MUST BOTH BE VARIABLES, NOT CONSTANTS. IREI0270
C     UPON RETURN, IREI IS SET TO..                                  IREI0280
C         =-1  IF NO LEGAL CHAR. IS FOUND (EXCEPT POSSIBLY BLANKS)   IREI0290
C              OR IF THE MAGNITUDE OF THE INPUT VALUE IS OUT OF RANGE. IREI0300
C         = 0  IF AN ILLEGAL CHAR. IS FOUND, BUT AFTER AT LEAST 1 LEGAL.IREI0310
C         IF ONE OF THESE CODES IS RETURNED, AND IF W GT 0,          IREI0320
C         THEN THE ILLEGAL CHAR. IS CHAR. C OF STRING S, AND         IREI0330
C         L IS A COUNT OF REMAINING CHARS IN S, INCL. THE ILL. CHAR. IREI0340
C         =+1  IF THERE ARE NO ILLEGAL CHARS IN THE INPUT FIELD.     IREI0350
C              IF W GT 0, C IS SET TO REFER TO THE 1ST CHAR AFTER THE IREI0360
C              INPUT FIELD, AND L IS A COUNT OF THE REMAINING CHAR. IN SIREI0370
C                                                                    IREI0380
      GS = W.GT.0                                                    IREI0390
      IL = L                                                         IREI0400
```

```
      IF (OS) IL=MIN0(L, )                                               IREI0410
C  SET UP CONVERSION                                                     IREI0420
      IK = 0                                                             IREI0430
      IF(IL.LE.0) GO TO 61                                               IREI0440
      IREI = 1                                                           IREI0450
      IXV = 0                                                            IREI0460
      IX = 1                                                             IREI0470
      ID = 0                                                             IREI0480
      FV = 0.0D                                                          IREI0490
      OX = .FALSE.                                                       IREI0500
C  SEARCH FOR MINUS SIGN (TO ALLOW PROPER HANDLING OF -0.XXX NUMBERS)    IREI0510
      IIS = +1                                                           IREI0520
      CALL RCHXTK(U,IK)                                                  IREI0530
      DO 11 I=1,IL                                                       IREI0540
      J = ICHXTK(S,U)                                                    IREI0550
      IF (J.EQ.IM) GO TO 13                                              IREI0560
      IF (J.NE.IB) GO TO 12                                              IREI0570
   11 CONTINUE                                                           IREI0580
      I = IL                                                             IREI0590
      GO TO 14                                                           IREI0600
   12 I = I-1                                                            IREI0610
      GO TO 14                                                           IREI0620
   13 IIS = -1                                                           IREI0630
   14 IK = IK+I                                                          IREI0640
      IL = IL-I                                                          IREI0650
C  CONVERT INTEGER PART, LESS SIGN                                       IREI0660
   15 LL = IL                                                            IREI0670
      II = INTI(S,IK,IL,LL,IIV)                                          IREI0680
      IF(IIV.LT.0) GO TO 60                                              IREI0690
      VV = IIV*IIS                                                       IREI0700
      IF(II.LE.0) GO TO 20                                               IREI0710
      ID = 0                                                             IREI0720
      GO TO 50                                                           IREI0730
C  SEARCH FOR DECIMAL POINT                                              IREI0740
   20 IF(KOM(S,IK,1,IP,1).NE.0) GO TO 38                                 IREI0750
   25 IK = IK+1                                                          IREI0760
      IL = IL-1                                                          IREI0770
      IF(IL.LE.0) GO TO 50                                               IREI0780
C  CONVERT FRACTION(IF PRESENT)                                          IREI0790
      IF = 0                                                             IREI0800
```

```
      IFV = 0
      LL = 0
      CALL RCHXTK(U,IK)
      DO 31 I=1,IL
      J = ICHXTK(S,U)
      IF(J.EQ.IB) GO TO 31
      IF(J.LT.IDO .OR. J.GT.I9) GO TO 35
      IF (LL.GE.MXI) GO TO 31
      IFV = IFV*10+J-ID
      LL = LL+1
   31 CONTINUE
      IF = 0
      I = IL+1
   35 IF(LL.LE.0) IF=-1
      I = I-1
      IK = IK+I
      IL = IL-I
      FV = IFV
      L1 = MOD(LL+50,10)+1
      L2 = (LL+50)/10
      FV = FV/T(L1)/TT(L2)
      IF(IIS.LT.0) FV=-FV
   34 IF(IF.GT.0) GO TO 50
      GO TO 40
   38 IF(II.LT.0) GO TO 60
      ID = D
C  SEARCH FOR 'E'
   40 IF(KOM(S,IK,1,IE,1).NE.0) GO TO 45
      GX = .TRUE.
   41 IK = IK+1
      IL = IL-1
      IF(IL.LE.0) GO TO 60
C  CONVERT EXPONENT
   45 LL = IL
      IX = INTI(S,IK,IL,LL,IXV)
      IF (IX.LT.0 .AND. (GX .OR. .NOT.QS)) GO TO 60
C  GENERATE FINAL VALUE AND CHECK SIZE
   50 VV = VV+FV
      IF(DABS(VV).LE.0.D0) GO TO 55
      IXV = IXV-ID
```

Reproduced from best available copy.

```
      IF(IXV.LT.-40 .OR. IXV.GE.+40) GO TO 60                            IREI1210
      L1 = MOD(IXV+50,10)+1                                              IREI1220
      L2 = (IXV+50)/10                                                   IREI1230
      VV = VV*T(L1)*TT(L2)                                               IREI1240
      IF(DABS(VV).LT.SMALL .OR. DABS(VV).GT.BIG) GO TO 60                IREI1250
   55 V = VV                                                             IREI1260
      IF(IX.LE.0) IREI=0                                                 IREI1270
      GO TO 70                                                           IREI1280
C  ERROR DETECTED                                                        IREI1290
   60 IREI = -1                                                          IREI1300
   61 V = 0.                                                             IREI1310
   70 IF(.NOT.PS) RETURN                                                 IREI1320
      C = IK                                                             IREI1330
      L = IL                                                             IREI1340
      RETURN                                                             IREI1350
      END                                                                IREI1360
```

```
C     **********
C     * ISCAN  *            JOHN ROUSE      IITRI       SEPT 1970
C     **********
C
      FUNCTION ISCAN(S1,C1,L1,TRT,C2,L2)
C
C     SCANS UP TO L1 CH., STARTING W/CH.C1 OF STRING S1.  PICKS
C     UP TOKENS, DEFINED BY USER, IN TRT TABLE.
C     A TOKEN IS 1) 1 OR MORE CH. W/ ZERO ENTRY IN TRT TABLE, OR
C                2) 1 CH., W/NONZERO TRT ENTRY
C     TOKEN TYPE IS INDICATED BY RETURNED VALUE OF ISCAN
C
C     ISCAN    = 0   TEXT (1 OR MORE ZERO-TRT-ENTRY CH.)
C     ISCAN    = 1   END OF STRING  (L1=0 ON INPUT.  C1 OUT=C1 IN)
C     ISCAN    .GT.1  DELIMITER  (NON-ZERO-TRT-ENTRY CH.)
C     C1       1ST CH. SCANNED,  ON OUTPUT, IS NEXT CH. TO BE SCANNED.
C     L1       NO. CH. TO BE SCANNED,  ON OUTPUT, IS NO.CH. LEFT.
C     TRT      64 WORD INTEGER TABLE, 1 ENTRY FOR EACH POSSIBLE CH.
C     C2       = INPUT VALUE OF C1
C     L2       = LEN. OF TOKEN.  (=0 IF END OF STRING)
C     STRING S IS PACKED (6 CHAR/WRD)
C
      INTEGER S1(1),TRT(64)
      INTEGER C1,C2,CN
C
C  L=0 SIGNALS END-OF-STRING
      IF(L1.GT.0) GO TO 1
      L2 = 0
      CN = C1
      ISCAN = 1
      GO TO 6
C  TRANSLATE AND TEST S1, USING PROVIDED TRT TABLE
    1 I = ITRT(S1,C1,L1,TRT,CN)
      IF(I) 4,5,4
    4 IF(CN.GT.C1+1) GO TO 7
C  DELIMITER FOUND
      L2 = 1
      ISCAN = IABS(I)
      GO TO 6
C  TEXT FOUND
```

```
    7 CN = CN-1                                                          ISCA0410
    5 L2 = CN-C1                                                         ISCA0420
      ISCAN =0                                                           ISCA0430
C   SET UP FOR NEXT SCAN, AND SET TOKEN LOCATION                         ISCA0440
    6 L1 = L1+L2                                                         ISCA0450
      C2 = C1                                                            ISCA0460
      C1 = CN                                                            ISCA0470
      RETURN                                                             ISCA0480
      END                                                                ISCA0490
```

```
C     **********
C     * ITRT *                 [illegible]        ITRT        SEPT 1970
C     **********
C
      FUNCTION ITRT(S1,C1,L1,TRT,CN)
C
C     EXAMINE UP TO L1 CH., STARTING W/CH.C1 OF STRING S1.
C     STOP WITH 1ST CH. WHOSE TRT ENTRY IS NONZERO).
C     RETURN WITH CN = NEXT CH. POS., AND
C     ITRT   =-     TOKEN OF TYPE N FOUND AT END OF STRING
C            = 0    NO NON-ZERO CH. ENCOUNTERED
C            = N    TOKEN OF TYPE N FOUND, MORE WILL FOLLOW
C     STRING S1 IS PACKED (6 CHAR/WRD)
C
      INTEGER S1(1),TRT(64),
      INTEGER C1,CN
C
      CALL RCHXTK(W,C1)
      IC = C1-1+L1
      DO 1 I=C1,IC
      J = ICHXTK(S1,I)+1
      ITRT = TRT(J)
      IF(ITRT.NE.0) GO TO 2
    1 CONTINUE
      CN = IC+1
      ITRT = 0
      RETURN
    2 CN = I+1
      IF(I.EQ.IC) ITRT=-ITRT
      RETURN
      END
```

```
C     **********
C     *  KO    *          JOHN ROUSE      IITRI       SEPT 1970
C     **********
C
      FUNCTION KOM (S1,C1,L,S2,C2)
      INTEGER S1(1),C1,W1,S2(1),C2,W2
C
C     COMPARE TWO STRINGS OVER L CHAR. (IF L.LE.0, RETURN A ZERO)
C     THE STRINGS ARE.. STRING S1, STARTING AT CHAR. C1, AND STRING S2,
C     STARTING AT CHAR. C2.  A VALUE BETWEEN -1 AND +1 IS RETURNED..
C          -1) S1.LT.S2      0) S1=S2      +1) S1.GT.S2
C     BOTH STRINGS S1 AND S2 ARE PACKED (I.E., 6 CHAR. PER WORD)
C
      KOM = 0
      IF(L.LE.0) RETURN
      CALL RCHXTK(W1,C1)
      CALL RCHXTK(W2,C2)
      DO 6 I=1,L
      IF(ICHXTK(S1,W1)-ICHXTK(S2,W2)) 1,6,2
    6 CONTINUE
      RETURN
    1 KOM = -1
      RETURN
    2 KOM = +1
      RETURN
      END
```

```
C     **********                                                        KOML0010
C     *  KOML  *          JOHN BRUSH       IITRI      MAY 1971          KOML0020
C     **********                                                        KOML0030
C                                                                       KOML0040
      FUNCTION KOML(S1,C1,L1,S2,C2,L2)                                  KOML0050
      INTEGER S1(1),C1,W1,S2(1),C2,W2                                   KOML0060
C                                                                       KOML0070
C     COMPARE TWO STRINGS OVER THE LENGTH OF THE SHORTER STRING.        KOML0080
C     THE TWO STRINGS ARE..                                             KOML0090
C          THE L1 CHAR. OF STRING S1, STARTING WITH CHAR. C1, AND       KOML0100
C          THE L2 CHAR. OF STRING S2, STARTING WITH CHAR. C2.           KOML0110
C     UPON RETURN, KOML IS SET TO..                                     KOML0120
C          =-2  IF S1 LT S2                                             KOML0130
C          =-1  IF S1 EQ S2 BUT L1 LT L2                                KOML0140
C          = 0  IF S1 EQ S2 AND L1 EQ L2                                KOML0150
C          =+1  IF S1 EQ S2 BUT L1 GT L2                                KOML0160
C          =+2  IF S1 GT S2                                             KOML0170
C                                                                       KOML0180
      L = L1                                                            KOML0190
      KOML = 0                                                          KOML0200
      IF (L1-L2) 4,5,3                                                  KOML0210
    3 L = L2                                                            KOML0220
      KOML = +1                                                         KOML0230
      GO TO 5                                                           KOML0240
    4 KOML = -1                                                         KOML0250
    5 IF (L.LE.0) RETURN                                                KOML0260
      CALL RCHXTK(W1,C1)                                                KOML0270
      CALL RCHXTK(W2,C2)                                                KOML0280
      DO 6 I=1,L                                                        KOML0290
      IF (ICHXTK(S1,W1)-ICHXTK(S2,W2)) 2,6,1                            KOML0300
    6 CONTINUE                                                          KOML0310
      RETURN                                                            KOML0320
    2 KOML = -2                                                         KOML0330
      RETURN                                                            KOML0340
    1 KOML = +2                                                         KOML0350
      RETURN                                                            KOML0360
      END                                                               KOML0370
```

```
      FUNCTION KOMW(S1,L,S2)                                            KOMW0010
C                                                                       KOMW0020
C     ***********                                                       KOMW0030
C     *  KOMW  *        JOHN ROUSE       IITRI      OCT 1970            KOMW0040
C     ***********                                                       KOMW0050
C                                                                       KOMW0060
      INTEGER S1(1),S2(1)                                               KOMW0070
C     COMPARE TWO ALPHABETIC STRINGS, S1 AND S2, EACH L WORDS LONG.     KOMW0080
C     UPON RETURN, KOMW IS SET TO..                                     KOMW0090
C         -1   IF S1 LT S2                                              KOMW0100
C          0   IF S1 EQ S2 (OR IF L LE 0, IE NULL STRINGS)              KOMW0110
C         +1   IF S1 GT S2                                              KOMW0120
C                                                                       KOMW0130
      KOMW = 0                                                          KOMW0140
      IF (L.LE.0) RETURN                                                KOMW0150
      DO 30 I=1,L                                                       KOMW0160
      IF (S1(I)) 10,5,15                                                KOMW0170
    5 IF (S2(I).EQ.0) GO TO 30                                          KOMW0180
      GO TO 35                                                          KOMW0190
   10 IF (S2(I).GE.0) GO TO 40                                          KOMW0200
      GO TO 20                                                          KOMW0210
   15 IF (S2(I)) 35,40,20                                               KOMW0220
C***  THIS STEP COUNTS ON NEG. VALUES IN 1- OR 2-COMPL FORM              KOMW0230
   20 IF (S1(I)-S2(I)) 35,30,40                                         KOMW0240
   30 CONTINUE                                                          KOMW0250
      RETURN                                                            KOMW0260
   35 KOMW = -1                                                         KOMW0270
      RETURN                                                            KOMW0280
   40 KOMW = +1                                                         KOMW0290
      RETURN                                                            KOMW0300
      END                                                               KOMW0310
```

```
C     ***********                                                       MOVE0010
C     * MOVE *          JO[illegible]       [illegible]       SEPT 197[illegible]   MOVE0020
C     ***********                                                       MOVE0030
      SUBROUTINE MOVE (S1,C1,L,S2,C2)                                   MOVE0040
      INTEGER S1(1),C1,[illegible],S2(1),C2,[illegible]                  MOVE0050
C                                                                       MOVE0060
C     MOVE L CHAR. FROM STRING S2, STARTING WITH CHAR. C2, TO STRING S1,MOVE0070
C     STARTING AT CHAR. C1. IF L.LE.0, NO MOVE TAKES PLACE.             MOVE0080
C     BOTH STRINGS S1 AND S2 ARE PACKED (I.E., 6 CHAR. PER WORD)        MOVE0090
C                                                                       MOVE0100
      IF(L.LE.0) RETURN                                                 MOVE0110
      CALL RCHXTK(C1,[illegible])                                       MOVE0120
      CALL RCHXTK(C2,C[illegible])                                      MOVE0130
      DO 1 I=1,L                                                        MOVE0140
    1 CALL SCHXTK(S1,[illegible])                                       MOVE0150
      RETURN                                                            MOVE0160
      END                                                               MOVE0170
```

```
C     ***********                                                     NOTA0010
C     * NOTARY *        JOHN ROUSE     IITRI      SEPT 1970            NOTA0020
C     ***********                                                     NOTA0030
C                                                                     NOTA0040
      SUBROUTINE NOTARY(FCN,N,M,XS,XE,MYY,YS,YE,CHAR)                 NOTA0050
C                                                                     NOTA0060
C     PLOTS N FUNCTIONS (N.LE.20) ON PRINTER                          NOTA0070
C     LIST 'CHAR' HOLDS PLOT CHARACTERS.. FCN NO. I IS PLOTTED USING  NOTA0080
C     CHAR(I).                                                        NOTA0090
C     FCN MUST BE OF THE FORM 'FCN(X,N,VAL)', AND MUST BE DECLARED    NOTA0100
C     EXTERNAL IN THE PROGRAM WHICH CALLS NOTARY.                     NOTA0110
C     FUNCTION VALUES ARE GENERATED BY FCN, AND STORED IN VECTOR      NOTA0120
C     VAL.  FCN VALUE I IS STORED IN VAL(I).  FCN IS CALLED M TIMES,  NOTA0130
C     AT X = XS, XS+DX, ...., XE.  (DX=(XE-XS)/(M-1))                 NOTA0140
C     THE Y AXIS INCREASES ALONG THE WIDTH OF THE PAGE, AND THE X AXIS NOTA0150
C     INCREASES DOWN THE PAGE.  THE Y RANGE IS YS THRU YE, IN MYY     NOTA0160
C     (MYY.LE.121) INCREMENTS.  OFF-SCALE FUNCTION VALUES ARE NOT PLOT- NOTA0170
C     TED.                                                            NOTA0180
      INTEGER POT(121),CHAR(N),DOT,BL                                 NOTA0190
      REAL VAL(20)                                                    NOTA0200
      DIMENSION IG(30)                                                NOTA0210
      LOGICAL QGRID                                                   NOTA0220
      DATA MG/30/                                                     NOTA0230
      DATA DOT/1H./,BL/1H /                                           NOTA0240
      IF(N.LE.20 .AND. N.GT.0 .AND. M.GT.0 .AND. MYY.GT.0) GO TO 201  NOTA0250
      WRITE(6,200) N,M,MYY                                            NOTA0260
  200 FORMAT(57H0******  ERROR FROM NOTARY.  20 FUNCTIONS ALLOWED. BUT NNOTA0270
     *>,I5/24X,16HOR.. NO. LINES (,I5,16H) OR NO. COLS. (,I5,10H) IN ERRNOTA0280
     *OR)                                                             NOTA0290
      RETURN                                                          NOTA0300
  201 MY = MINO(MYY,121)                                              NOTA0310
      DX = (XE-XS)/FLOAT(M-1)                                         NOTA0320
      DY = (YE-YS)/FLOAT(MY-1)                                        NOTA0330
      MGJ = MAXO(2,MINO(MY/20+1,MG-2))                                NOTA0340
      MGI = MG-MGJ                                                    NOTA0350
      CALL PPGRID(YS,YE,MY,NGJ,MGJ,IG,20)                             NOTA0360
      CALL PPGRID(XS,XE,M,NGI,MGI,IG(NGJ+1),12)                       NOTA0370
      NGI = NGI+NGJ                                                   NOTA0380
      KGI = NGJ+1                                                     NOTA0390
      SC = AMAX1(ABS(XS),ABS(XE))                                     NOTA0400
```

```
      IF(SC.GT.0.) SC=ALOG10(SC)
      IF(SC.LT.0.) SC=SC-3.
      SC = AINT(SC/3.)*3.
      SC = 10.**(-AINT(SC))
      WRITE(6,202) YS,SC,YE
  202 FORMAT(11X,6HYMIN =,E12.4,31X,10HX MULT. BY,E12.4,32X,6HYMAX =,
     * E12.4/)
      DSC = DX*SC
      XSC = XS*SC
      X=XS
      DO 102 I=1,M
      QGRID=.FALSE.
      IF(KGI.GT.NGI) GO TO 110
      IF(I.LT.IG(KGI)) GO TO 110
      QGRID = .TRUE.
      KGI = KGI+1
  110 KGJ = 1
      DO 101 J=1,MY
      IF(QGRID) GO TO 111
      IF(KGJ.GT.NGJ) GO TO 112
      IF(J.LT.IG(KGJ)) GO TO 112
      KGJ = KGJ+1
  111 POT(J) = DOT
      GO TO 101
  112 POT(J) = BL
  101 CONTINUE
      CALL FCN(X,N,VAL)
      DO 103 J=1,N
      IY = (VAL(J)-YS)/DY+1.
      IF(IY.GE.1 .AND. IY.LE.MY) POT(IY)=CHAR(J)
  103 CONTINUE
      WRITE(6,151) XSC,(POT(J),J=1,MY)
  151 FORMAT(1X,F8.2,2X,121A1)
      XSC = XSC+DSC
  102 X=X+DX
      RETURN
      END
```

```
C     ***********
C     * PPGRID *           JOHN HOUSE       IITRI        SEPT 1970
C     ***********
C
      SUBROUTINE PPGRID(QMIN,QMAX,NG,NQ,MG,IG,JQ)
      DIMENSION IG(1)
C
C     COMPUTE GRID LINE POSITIONS FOR PRINTER PLOTTER.
C     AXIS RANGES FROM QMIN TO QMAX.  UP TO MG GRID LINES MAY BE PRO-
C     DUCED, THE ACTUAL NUMBER IS RETURNED IN NG.  VECTOR IG HOLDS THE
C     GRID LINE INDICES.  IG(1) WILL ALWAYS BE 1, AND IG(NG) WILL ALWAYS
C     EQUAL NQ, THE NUMBER OF CHAR. POS. OR LINES USED BY THE AXIS.
C     GRID LINES ARE CHOSEN TO BE AT LEAST JQ/4 COLS. OR LINES FROM
C     EITHER END OF THE AXIS, AND FROM JQ TO 2*JQ COLS/LINES APART FROM
C     EACH OTHER.  GRID LINES ARE CHOSEN  AT 'ROUND' VALUES OF AXIS
C     VARIABLE
C
      NG = 1
      IG(1) = 1
      IF(MG-2) 54,50,43
   43 QR = QMAX-QMIN
      IDUM = NQ/JQ+1
      DG = QR/FLOAT(IDUM)
      DQ = QR/FLOAT(NQ-1)
      DG = ALOG10(DG)
      IF(DG.LT.0.) DG=DG-1.
      DG = 10.**AINT(DG)
      DUM = FLOAT(NQ)*DG/QR
   41 IF(DUM.GE.FLOAT(JQ)) GO TO 40
      DUM = DUM*2.
      DG = DG*2.
      GO TO 41
   40 IF (DUM.LE.FLOAT(2*JQ)) GO TO 42
      DUM = DUM*0.5
      DG = DG*0.5
      GO TO 40
   42 JDUM = JQ/4
      IDUM = NQ-JDUM
      JDUM = JDUM+1
      DUM = AINT(QMIN/DG)*DG
```

```
      IF(Q[illegible].LT.0.) D[illegible]=D[illegible]-[illegible]            PPGR0410
      IF(J[illegible].LT.[illegible]) GO TO 52                         PPGR0420
   51 J = ([illegible]-J[illegible])/2+1,                         PPGR0430
      IF(J.LE.J[illegible]) GO TO 52                              PPGR0440
      IF(J.GE.[illegible]) GO TO 5[illegible]                     PPGR0450
      NG = [illegible]+1                                          PPGR0460
      IF(NG.GE.[illegible]) GO TO [illegible]                     PPGR0470
      IG([illegible]) = J                                         PPGR0480
   52 [illegible] = [illegible]+[illegible]                       PPGR0490
      GO TO 51                                                    PPGR0500
   50 NG = [illegible]+1                                          PPGR0510
   53 IG([illegible]) = N[illegible]                              PPGR0520
   54 RETURN                                                      PPGR0530
      END                                                         PPGR0540
```

```
.         CHARACTER STORE AND EXTRACT ROUTINES
.
.         RETNA
.
$(1).
.
.         SET POINTER 'K' TO CHAR. 'C' OF STRING.
.
.         FORTRAN CALL IS..
.                   CALL RCHXTK( ,K)
.
RCHXTK*   LA,U16     A0,0              . CLEAR TOP OF DIVIDEND
          LA         A1,*1,R11         . LOAD BOTTOM (ST. CHAR. COUNT)
          AA,U16     A1,5              . PLUS 5
          DI,U16     A0,6              . DIVIDE BY 6 TO GET..
          SA,Q1      A0,*2,R11         . ST. WORD NO. (QUOTIENT), AND
          SA,Q2      A1,*2,R11         . ST. CHAR. NO. WITHIN WORD (REM.)
          J          3,R11
.
.         SET NEXT CHAR. IN STRING 'S' TO BIT PATTERN IN THE LOWER 6
.         BITS OF 'K'.  'W' IS STRING PTR.
.
.         FORTRAN CALL IS..
.                   CALL SCHXTK(S, ,K)
.
SCHXTK*   LA,Q1      A3,*1,R11         . GET CURR. WORD NO.
          LA,Q2      A2,*1,R11         . GET CURR. CHAR. NO.
          LA         A1,0,R11          . GET ADDR OF START OF STRING
          AA         A1,A3             . COMPUTE ADDR OF CURRENT WORD..
          ANA,U16    A1,1              .    IS START-1+WORD NO.
          LA         A0,*2,R11         . GET CHARACTER TO BE STORED,
          EX         STORCH,A2         . EXECUTE PROPER CHAR. STORE INSTR.
          LMJ        A1,NEXTCH         . JUMP TO PTR UPDATE ROUTINE
          J          4,R11             . AND EXIT
.
.        RETRIEVE NEXT CHAR. FROM STRING 'S'.  'W' IS STRING PTR.
.
.         FORTRAN CALL IS..
.                   K = ICHXTK(S,W)
.
```

```
ICHXTK*   LA,01      A3,*1,R11      . GET CURR. WORD NO.                       RCHX0410
          LA,02      A2,*1,R11      . AND CURR. CHAR. NO.                      RCHX0420
          LA         A1,0,R11       . GET ADDR OF START OF STRING              RCHX0430
          AA         A1,A3          . COMPUTE ADDR OF CURRENT WORD..           RCHX0440
          ANA,016    A1,1           .    IS START-1+WORD NO.                   RCHX0450
          EX         LOADCH,A2      . EXECUTE PROPER CHAR. LOAD INSTR.         RCHX0460
          LMJ        A1,NEXTCH      . JUMP TO PTR UPDATE ROUTINE               RCHX0470
          J          3,R11          . AND EXIT                                 RCHX0480
.                                                                              RCHX0490
NEXTCH    AA,016     A2,1           . ADD 1 TO CHAR NO                         RCHX0500
          TLE,016    A2,6           . SKIP NI IF NEW CHAR NO GE 6              RCHX0510
          J          SAMEWD         . ELSE ARE IN SAME WORD AS BEFORE          RCHX0520
          AA,016     A3,1           . NEW WORD.  SET CHAR NO TO ZERO, ANDRCHX0530
          SA         A3,*1,R11      . UP WORD NO BY 1                          RCHX0540
          J          0,A1                                                      RCHX0550
SAMEWD    SA,02      A2,*1,R11      . SAME WORD.  STORE NEW CHAR NO            RCHX0560
          J          0,A1                                                      RCHX0570
.                                                                              RCHX0580
.         SIMILAR TO SCHXTK, BUT CHARACTERS ARE STORED FROM RIGHT TO          RCHX0590
.         LEFT.                                                                RCHX0600
.                                                                              RCHX0610
.         FORTRAN CALL IS..                                                    RCHX0620
.                    CALL PCHXTK(S,W,K)                                        RCHX0630
.                                                                              RCHX0640
PCHXTK*   LA,01      A3,*1,R11      . GET CURR. WORD NO.                       RCHX0650
          LA,02      A2,*1,R11      . GET CURR. CHAR. NO.                      RCHX0660
          LA         A1,0,R11       . GET ADDR OF START OF STRING              RCHX0670
          AA         A1,A3          . COMPUTE ADDR OF CURRENT WORD..           RCHX0680
          ANA,016    A1,1           .    IS START-1+WORD NO.                   RCHX0690
          LA         A0,*2,R11      . GET CHARACTER TO BE STORED.              RCHX0700
          EX         STORCH,A2      . EXECUTE PROPER CHAR. STORE INSTR.        RCHX0710
          LMJ        A1,PREVCH      . JUMP TO PTR UPDATE ROUTINE               RCHX0720
          J          4,R11          . AND EXIT                                 RCHX0730
.                                                                              RCHX0740
.         SIMILAR TO ICHXTK, BUT CHARACTERS ARE RETRIEVED FROM RIGHT TO RCHX0750
.         LEFT.                                                                RCHX0760
.                                                                              RCHX0770
.         FORTRAN CALL IS..                                                    RCHX0780
.                    K = JCHXTK(S,W)                                           RCHX0790
.                                                                              RCHX0800
```

```
JCHXTK*  LA,01     A3,*1,R1    . GET CURR. WORD NO.                     RCHX0810
         LA,02     A2,*1,R1    . AND CURR. CHAR. NO.                    RCHX0820
         LA        A1,,R1      . GET ADDR OF START OF STRING            RCHX0830
         AA        A1,A3       . COMPUTE ADDR OF CURRENT WORD..         RCHX0840
         ANA,U     A1,1        .    IS START-1+WORD NO.                 RCHX0850
         EX        LOADCH,A2   . EXECUTE PROPER CHAR. LOAD INSTR.       RCHX0860
         LMJ       X1,PREVCH   . JUMP TO PTR UPDATE ROUTINE             RCHX0870
         J         3,X11                                                RCHX0880
.                                                                       RCHX0890
PREVCH   ANA,U14   A2,1        . SUBTRACT 1 FROM CHAR. NO.              RCHX0900
         TLE,U14   A2,0        . SKIP NI IF NEW CHAR. NO. GE 0.         RCHX0910
         J         PREVWD      . ELSE ARE STARTING PREVIOUS WORD        RCHX0920
         SA,02     A2,*1,R1    . SAME WORD.  STORE NEW CHAR. NO.        RCHX0930
         J         0,X1        . AND EXIT                               RCHX0940
PREVWD   ANA,U14   A3,1        . PREVIOUS WORD.  SET CHAR. NO. TO 5,    RCHX0950
         LXI,U14   A3,5        . PACK INTO ONE REGISTER                 RCHX0960
         SA        A3,*1,R1    . AND STORE INTO PTR WORD                RCHX0970
         J         0,X1        . AND EXIT                               RCHX0980
.                                                                       RCHX0990
$(1).                                                                   RCHX1000
.                                                                       RCHX1010
.                      STORE CHAR 0 THRU CHAR 5                         RCHX1020
STORCH   SA,015    A0,0,A1                                              RCHX1030
         SA,014    A0,0,A1                                              RCHX1040
         SA,013    A0,0,A1                                              RCHX1050
         SA,012    A0,0,A1                                              RCHX1060
         SA,011    A0,0,A1                                              RCHX1070
         SA,010    A0,0,A1                                              RCHX1080
.                      LOAD CHAR 0 THRU CHAR 5                          RCHX1090
LOADCH   LA,015    A0,0,A1                                              RCHX1100
         LA,014    A0,0,A1                                              RCHX1110
         LA,013    A0,0,A1                                              RCHX1120
         LA,012    A0,0,A1                                              RCHX1130
         LA,011    A0,0,A1                                              RCHX1140
         LA,010    A0,0,A1                                              RCHX1150
         END                                                            RCHX1160
```

```
      SUBROUTINE SOJAC(FCN,XMIN,XMAX,YMIN,YMAX,FMIN,FMAX,IYR,INT,NF,L)  SOJA0010
C                                                                       SOJA0020
C     **********                                                        SOJA0030
C     * SOJAC  *          JOHN HOUSE      IITRI        SEPT 1970        SOJA0040
C     **********                                                        SOJA0050
C                                                                       SOJA0060
C***  X-VALUES ARE SCALED BY APPROPRIATE POWER OF TEN.                  SOJA0070
C     PLOTS CONTOUR MAP OF FCN(X,Y) ON PRINTER.                         SOJA0080
C                                                                       SOJA0090
C     NOTE.. IN CALLING ROUTINE 'FCN' MUST APPEAR IN AN 'EXTERNAL'      SOJA0100
C     STATEMENT.                                                        SOJA0110
C                                                                       SOJA0120
C     X RANGES BETWEEN XMIN AND XMAX.  X INCREASES DOWN THE PAGE.       SOJA0130
C                                                                       SOJA0140
C     Y RANGES BETWEEN YMIN AND YMAX.  Y INCREASES ACROSS THE PAGE.     SOJA0150
C                                                                       SOJA0160
C     F(X,Y) IS EXPECTED TO RANGE BETWEEN FMIN AND FMAX.                SOJA0170
C     L(3) IS PLOTTED IF A FUNCTION VALUE IS .GE. 1.E+38 (I.E. UNDE-    SOJA0180
C     FINED.)  L(4) IS PLOTTED FOR VALUES BELOW FMIN.  L(NF+4) FOR ABOVESOJA0190
C     FMAX.  THE REGION BETWEEN FMIN AND FMAX IS DIVIDED INTO NF EQUAL  SOJA0200
C     INTERVALS.  INTERVAL K IS PLOTTED WITH SYMBOL L(K+4).             SOJA0210
C                                                                       SOJA0220
C     TO HELP SMOOTH THE CONTOURS, THIS PROCEDURE IS FOLLOWED..  IF THE SOJA0230
C     FUNCTION VALUE AT A POINT BE WITHIN 0.05 INTERVAL WIDTH FROM THE  SOJA0240
C     EDGE OF AN INTERVAL, IT WILL BE PLOTTED WITH A BORDER CHARACTER.  SOJA0250
C     IF THE PLOT POSITION ABOVE THE CURRENT POSITION  (I.E., SAME Y,   SOJA0260
C     LOWER X) BE BLANK, THE BORDER CHARACTER IS L(2).  OTHERWISE IT IS SOJA0270
C     L(1).  THIS PROCEDURE IS NOT USED IF L(1) AND L(2) ARE BLANK.     SOJA0280
C                                                                       SOJA0290
C     THE PLOT IS IYR PRINT POSITIONS WIDE.  THE LENGTH OF THE PLOT IS  SOJA0300
C     WHATEVER IT HAS TO BE TO KEEP THE UNIT OF LENGTH THE SAME IN THE  SOJA0310
C     X AND Y DIRECTIONS.                                               SOJA0320
C     IF INT = 0, SOJAC RETURNS AFTER COMPUTING THESE QUANTITIES, FOUND SOJA0330
C     IN COMMON /CSOJAC/.,  DX, DY THE AMOUNT X AND Y ARE STEPPED       SOJA0340
C     BETWEEN EVALUATION POINTS, AND IXR, THE NUMBER OF POINTS IN THE   SOJA0350
C     X DIRECTION.  THERE ARE IYR*IXR POINTS PLOTTED IN ALL.            SOJA0360
C     IF INT.GT.0, A PLOT IS GENERATED.                                 SOJA0370
C                                                                       SOJA0380
C                             THE Y RANGE AND THE WIDTH OF AN INTERVAL  SOJA0390
C     ARE PLOTTED.  THE VALUE OF X IS PRINTED ALONG THE LEFT MARGIN.    SOJA0400
```

```
C                                                                       SOJA0410
      COMMON /GSPOJAC/D) ,IX,IXH,IYY,I,J                                SOJA0420
C                                                                       SOJA0430
      INTEGER POT,DOT,PL,IG(30)                                         SOJA0440
      LOGICAL OFILL,OGRID                                               SOJA0450
      DIMENSION PCT(125,2),L(1)                                         SOJA0460
      DATA DOT,BL,MG/1H.,1H ,30/                                        SOJA0470
C                                                                       SOJA0480
      IYY = IYR                                                         SOJA0490
      IF(IYR.LE.0.OR.IYR.GT.121) IYY = 121                              SOJA0500
      OFILL = L(1).NE.BL .OR. L(2).NE.BL                                SOJA0510
      MGJ = MAX0(2,MIN0(IYY/20+1,MG-2))                                 SOJA0520
      MGI = MG-MGJ                                                      SOJA0530
      CALL PPGRID(YMIN,YMAX,IYY,IG ,MGJ,IG,20)                          SOJA0540
C                     SET UP X AND Y SCALES, GRID LINE INTERVALS        SOJA0550
      YR = YMAX-YMIN                                                    SOJA0560
      XR = XMAX-XMIN                                                    SOJA0570
      DY = YR/FLOAT(IYY-1)                                              SOJA0580
      DX=DY*1.666667                                                    SOJA0590
      DX2 = DX/2.                                                       SOJA0600
      IXR = XR/DX+1.5                                                   SOJA0610
      CALL PPGRID(XMIN,XMAX,IXR,MGI,NGI,IG(NGJ+1),12)                   SOJA0620
      NGI = NGI+NGJ                                                     SOJA0630
      KGI = NGJ+1                                                       SOJA0640
C                    IF INT .LE.0, ASSUME THAT ONLY SCALE DATA WAS WANTEDSOJA0650
      IF(INT.LE.0) RETURN                                               SOJA0660
C                    SET UP FUNCTION RANGE, AND CONTOUR INTERVAL        SOJA0670
      FR = FMAX-FMIN                                                    SOJA0680
      FNF = NF                                                          SOJA0690
      DF = FR/FNF                                                       SOJA0700
      XSC = AMAX1(ABS(XMIN),ABS(XMAX))                                  SOJA0710
      IF(FR.GT.0. .AND. XSC.GT.0.) GO TO 130                            SOJA0720
      WRITE(6,131) FMIN,FMAX                                            SOJA0730
  131 FORMAT(19HONULL PLOT,  FMIN =,E15.7,8H, FMAX =,E15.7)             SOJA0740
      RETURN                                                            SOJA0750
  130 XSC = ALOG10(XSC)                                                 SOJA0760
      IF(XSC.LT.0.) XSC=XSC-3.                                          SOJA0770
      XSC = AINT(XSC/3.)*3.                                             SOJA0780
      XSC = -AINT(XSC)                                                  SOJA0790
      XSC = 10.**XSC                                                    SOJA0800
```

```
      DXS = DX*XSC                                                      SOJA0810
C                        PRINT PLOT ...                                 SOJA0820
      WRITE(6,152) YMIN,DF,XSC,YMAX                                     SOJA0830
  152 FORMAT(11X,6HYMIN =,E12.4,12X,16HCONTOUR INTERVAL =,E12.4,12X,    SOJA0840
     * 10HX MULT. BY,E8.2,12X,6HYMAX =,E12.4/)                          SOJA0850
      X = XMIN                                                          SOJA0860
      XS = X*XSC                                                        SOJA0870
      II = 2                                                            SOJA0880
      DO 102 I=1,IX                                                     SOJA0890
      KK  = II                                                          SOJA0900
      II = 3-II                                                         SOJA0910
      DGRID = .FALSE.                                                   SOJA0920
      IF(KGI.GT.NGI) GO TO 108                                          SOJA0930
      IF(I.LT.IG(KGI)) GO TO 108                                        SOJA0940
      DGRID = .TRUE.                                                    SOJA0950
      KGI = KGI+1                                                       SOJA0960
  108 KGJ = 1                                                           SOJA0970
      Y = YMIN                                                          SOJA0980
      DO 103 J=1,IYY                                                    SOJA0990
      POT(J,II) = DOT                                                   SOJA1000
      IF (DGRID) GO TO 103                                              SOJA1010
      IF(KGJ.GT.NGJ) GO TO 101                                          SOJA1020
      IF(J.LT.IG(KGJ)) GO TO 101                                        SOJA1030
      KGJ = KGJ+1                                                       SOJA1040
      GO TO 103                                                         SOJA1050
  101 VAL = FCN(X,Y)                                                    SOJA1060
C                        CONVERT FUNCTION VALUE INTO PLOT SYMBOL        SOJA1070
      IF(VAL.GE.1.E+38) GO TO 114                                       SOJA1080
      TV = (VAL-FMIN)/DF+5.                                             SOJA1090
      ATV = AINT(TV+0.05)                                               SOJA1100
      IX = TV                                                           SOJA1110
      IF(DFILL .AND. IX.GE.5 .AND. IX.LE.NF+4 .AND. ABS(TV-ATV).LE.0.05)SOJA1120
     * GO TO 110                                                        SOJA1130
      IX = MIN0(NF+5,MAX0(4,IFIX(TV)))                                  SOJA1140
      POT(J,II) = L(IX)                                                 SOJA1150
      GO TO 111                                                         SOJA1160
  110 POT(J,II) = L(1)                                                  SOJA1170
      IF(I.EQ.1) GO TO 111                                              SOJA1180
      IF(POT(J,KK).NE.BL) POT(J,II)=L(2)                                SOJA1190
      GO TO 111                                                         SOJA1200
```

```
  114 POT(J,II) = L(3)                                                    SOJA1210
  111 CONTINUE                                                            SOJA1220
  103 Y=Y+DY                                                              SOJA1230
C                   PRINT A LINE OF PLOT SYMBOLS, AND CORRES. X VALUE     SOJA1240
  161 WRITE(6,151) XS,(POT(J,II),J=1,IYY)                                 SOJA1250
  151 FORMAT(1X,F8.2,2X,121A1)                                            SOJA1260
      GO TO 112                                                           SOJA1270
  112 XS = XS+DXS                                                         SOJA1280
  102 X=X+DX                                                              SOJA1290
      RETURN                                                              SOJA1300
      END                                                                 SOJA1310
```

```
      LOGICAL FUNCTION SYNTAX(S1,C1,L1,X,IX,TRT,U,STK,NS,W)            SYNT0010
C                                                                       SYNT0020
C     **********                                                        SYNT0030
C     * SYNTAX *        JOHN ROUSE      IITRI       SEPT 197           SYNT0040
C     **********                                                        SYNT0050
C                                                                       SYNT0060
C     SCAN STRING, USING A LIST OF SYNTAX RULES.  A USER EXIT IS CALLED SYNT0070
C     AS THE RULES DICTATE.  UPON RETURN, SYNTAX = .TRUE. IF STRING IS  SYNT0080
C     CONSISTENT WITH RULES, OR SYNTAX=.FALSE. IF NOT.                  SYNT0090
C                                                                       SYNT0100
C     S1        STRING TO BE EXAMINED.                                  SYNT0110
C     C1        STARTING CHAR. TO BE EXAMINED.  S1 MUST BE A VARIABLE.  SYNT0120
C               BECAUSE UPON RETURN C1 = 1ST CHAR. BEYOND END OF SCAN.  SYNT0130
C     L1        NO. OF CHARS. TO BE EXAMINED.  L1 MUST BE A VARIABLE,   SYNT0140
C               BECAUSE UPON RETURN L1 = NO. CHAR. LEFT UNEXAMINED.     SYNT0150
C               (IF THE STRING IS OK, C1 OUT = C1 IN + L1 IN, AND       SYNT0160
C               L1 OUT = 0                                              SYNT0170
C     X         LIST OF RULES WHICH DESCRIBE SYNTAX                     SYNT0180
C     IX        1ST RULE NUMBER -- I.E., THE STARTING POINT             SYNT0190
C     TRT       ADDRESS OF CHAR. ATTRIBUTE TABLE -- ONE ENTRY FOR EACH  SYNT0200
C               CHAR.  VALUE OF EACH ENTRY IS CLASS TO WHICH CH. BELONGS.SYNT0210
C     U         USER-PROVIDED ROUTINE, CALLED AS DIRECTED BY THE RULES. SYNT0220
C               MUST BE OF THE FORM..                                   SYNT0230
C                    LOGICAL FUNCTION U(S1,C2,L2,K,W)                   SYNT0240
C               WHERE C2 AND L2 DELIMIT A PARTICULAR FIELD AND K IS A   SYNT0250
C               RULES-DETERMINED ACTION CODE.                           SYNT0260
C     STK       TEMPORARY STORAGE VECTOR.                               SYNT0270
C     NS        LENGTH OF STK.                                          SYNT0280
C     W         GENERAL VECTOR FOR USE BY USER-SUPPLIED ROUTINE.  FIRST SYNT0290
C               -SEVERAL- CELLS ARE FOR STORAGE AS DIRECTED BY THE RULES.SYNT0300
C                                                                       SYNT0310
C     SUBPROGRAMS CALLED..                                              SYNT0320
C     ISCAN     ISOLATES CHAR. GROUPS (TOKENS) IN INPUT STREAM          SYNT0330
C     U         USER-SUPPLIED ROUTINE (SEE ABOVE)                       SYNT0340
C     SYNTXF    SPECIAL ROUTINE TO RETRIEVE FIELDS OF RULES             SYNT0350
C     RCHXTK, ICHXTK    BASIC CHARACTER EXTRACTION ROUTINES             SYNT0360
C                                                                       SYNT0370
C***  TRT IS A TRANSLATE TABLE (SEE ABOVE), AS REQUIRED BY ISCAN.       SYNT0380
      INTEGER S1(1),TRT(1),X(1),W(1),C1,STK(1)                          SYNT0390
      INTEGER V(8),S,B,C2,C,D0,A,A1,A2,T,F                              SYNT0400
```

```
      LOGICAL U                                                         SYNT0410
      LOGICAL OPF5(3,6)                                                 SYNT0420
      EQUIVALENCE (Y(1),O),(Y(2),OO),(Y(3),A1),(Y(4),A2),(Y(5),A),      SYNT0430
     *            (Y(6),K),(Y(7),T),(Y(8),F)                            SYNT0440
      DATA OPF5/.TRUE.,3*.FALSE.,.TRUE.,.FALSE.,2*.TRUE.,3*.FALSE.,     SYNT0450
     * 2*.TRUE.,.FALSE.,.TRUE.,.FALSE.,2*.TRUE./                        SYNT0460
C***  NWDR IS NO. 'WORDS' PER SYNTAX RULE                               SYNT0470
      DATA NWDR/2/                                                      SYNT0480
C                                                                       SYNT0490
C  EMPTY STACK, SET UP SCAN, POINT TO INITIAL RULE                      SYNT0500
C                                                                       SYNT0510
      S = 0                                                             SYNT0520
      LL = L1+1                                                         SYNT0530
      R = (IX-1)*NWDR+1                                                 SYNT0540
      I = -1                                                            SYNT0550
C                                                                       SYNT0560
C  SCAN -- PICK UP ONE TOKEN (GROUP) FROM STRING.  EITHER TEXT OR DELIM.SYNT0570
C                                                                       SYNT0580
    1 ISAVE = I                                                         SYNT0590
      I = ISCAN(S1,C1,L1,TRT,C2,L2)                                     SYNT0600
C                                                                       SYNT0610
C  APPLY A RULE TO LATEST TOKEN                                         SYNT0620
C                                                                       SYNT0630
    2 CALL SYNTXF(9,Y,X(R))                                             SYNT0640
      GO TO (100,200,300,400,500,505,600,601,700,800,810,850,860),O     SYNT0650
    9 SYNTAX = .FALSE.                                                  SYNT0660
    8 IF(O.EQ.2) RETURN                                                 SYNT0670
      C1 = C1-L2                                                        SYNT0680
      L1 = L1+L2                                                        SYNT0690
      RETURN                                                            SYNT0700
C                                                                       SYNT0710
C  O=1 -- PROCEDURE CALL                                                SYNT0720
C                                                                       SYNT0730
  100 S = S+2                                                           SYNT0740
      IF(S.GT.NS) GO TO 9                                               SYNT0750
C  STACK STRING INFO AND RETURN ADDR.                                   SYNT0760
      STK(S-1) = C1-L2                                                  SYNT0770
      STK(S) = R                                                        SYNT0780
C  RULE-PROCEDURE ADDRESS IS KEPT AS ARGUMENT OF CALL                   SYNT0790
      R = A                                                             SYNT0800
```

```
      GO TO 2
C
C  O=2 -- EXAMINE A TOKEN FOR (NO) MEMBERSHIP IN A SET OF TOKEN TYPES
C
  200 GO TO (300,210,220,230,240),OO
  210 SYNTAX = I .EQ. A1
      GO TO 209
  220 SYNTAX = I .NE. A1
      GO TO 209
  230 SYNTAX = A1 .LE. I .AND. I.LE. A2
      GO TO 209
  240 SYNTAX = I .LT. A1 .OR. I .GT. A2
C
C  CHECK VALUE OF 'SYNTAX' -- DETERMINES SUCCESS/FAILURE
C
  209 IF(SYNTAX) GO TO 290
C
C  SYNTAX EXAMINATION (OR OTHER TEST) FAILS
C
  280 CALL SYNTXF(8,Y,X(B))
      B = F
      IF (B.GT.0) GO TO 2
C  IF NO F-ADDR., POP STACK (RETURN FROM RULE-PROC.) IF STACK NON-EMPTY
  282 IF(S.LE.0) GO TO 9
      B = STK(S)
      C1 = STK(S-1)
      L1 = LL-C1
  283 S = S-2
      CALL SYNTXF(8,Y,X(B))
      B = F
      IF(B.LE.0) GO TO 282
      GO TO 1
C
C  SYNTAX EXAMINATION (OR OTHER TEST) SUCCEEDS
C
  290 CALL SYNTXF(10,Y,X(B))
      IF (K.LE.0) GO TO 291
      IF (.NOT. U(S1,C2,L2,K,W)) GO TO 280
  291 B = T
C  B IS A SUCCESSFUL BRANCH ADDRESS
```

```
  295 IF(B.LE.0) GO TO 292                                              SYNT1210
      IF(D.EQ.2) GO TO 1                                                SYNT1220
      GO TO 2                                                           SYNT1230
C  IF NO T-ADDR., POP STACK IF NON-EMPTY (I.E. RETURN FROM RULE-PROC.)  SYNT1240
  292 IF(S.LE.0) GO TO R                                                SYNT1250
      B = STK(S)                                                        SYNT1260
      S = S-2                                                           SYNT1270
      GO TO 295                                                         SYNT1280
C                                                                       SYNT1290
C  O=3 -- FORCE GOOD-SYNTAX RESPONSE (UNLESS USER ROUTINE RETURNS FALSE)SYNT1300
C                                                                       SYNT1310
  300 SYNTAX = .TRUE.                                                   SYNT1320
      GO TO 295                                                         SYNT1330
C                                                                       SYNT1340
C  O=4 -- FORCE BAD-SYNTAX RESPONSE                                     SYNT1350
C                                                                       SYNT1360
  400 SYNTAX = .FALSE.                                                  SYNT1370
      GO TO 285                                                         SYNT1380
C                                                                       SYNT1390
C  O=5 -- COMPARE W(A1) WITH A2                                         SYNT1400
C  O=6 -- COMPARE W(A1) WITH W(A2)                                      SYNT1410
C                                                                       SYNT1420
  505 A2 = W(A2)                                                        SYNT1430
  500 IF(W(A1)-A2) 501,502,503                                          SYNT1440
  501 J = 1                                                             SYNT1450
      GO TO 510                                                         SYNT1460
  502 J = 2                                                             SYNT1470
      GO TO 510                                                         SYNT1480
  503 J = 3                                                             SYNT1490
  510 IF (OPE5(J,OO)) GO TO 300                                         SYNT1500
      GO TO 400                                                         SYNT1510
C                                                                       SYNT1520
C  O=7 -- SET W(A1) TO SOME FCN OF W(A1) AND A2                         SYNT1530
C  O=8 -- SET W(A1) TO SOME FCN OF W(A1) AND W(A2)                      SYNT1540
C                                                                       SYNT1550
  601 A2 = W(A2)                                                        SYNT1560
  600 GO TO (611,613,615,617,619,621,623,625,627,629,631,633,635,637,   SYNT1570
     *        639,641,643,645),OO                                       SYNT1580
  611 W(A1) = A2                                                        SYNT1590
      GO TO 300                                                         SYNT1600
```

```
613 W(A1) = -A2
    GO TO 300
615 W(A1) = IABS(A2)
    GO TO 300
617 W(A1) = -IABS(A2
    GO TO 300
619 W(A1) = W(A1)+A2
    GO TO 300
621 W(A1) = W(A1)-A2
    GO TO 300
623 W(A1) = W(A1)*A2
    GO TO 300
625 IF(A2.EQ.0) GO TO 400
    W(A1) = W(A1)/A2
    GO TO 300
627 IF (A2.EQ.0) GO TO 400
    W(A1) = (W(A1)-1+A2)/A2
    GO TO 300
629 IF (A2.LE.0) GO TO 400
    W(A1) = MOD(W(A1),A2)
    GO TO 300
631 IF (A2.LT.0) GO TO 400
    W(A1) = W(A1)**A2
    GO TO 300
633 W(A1) = MIN0(W(A1),A2)
    GO TO 300
635 W(A1) = MAX0(W(A1),A2)
    GO TO 300
637 W(A1) = ISIGN(W(A1),A2)
    GO TO 300
639 SYNTAX = A2.LE.0
    GO TO 209
641 SYNTAX = W(A1).GT.0 .OR. A2.GT.0
    GO TO 209
643 SYNTAX = W(A1).GT.0 .AND. A2.GT.0
    GO TO 209
645 SYNTAX = W(A1).GT.0 .AND. A2.LE.0 .OR. W(A1).LE.0 .AND. A2.GT.0
    GO TO 209
C
C  0=9 -- SET W(A1) TO SOME 'SPECIAL VALUE'
```

```
C                                                                       SYNT2010
  700 GO TO (705,710,720,730,740,750,760,770),30                        SYNT2020
C  SET W(A1) TO VALUE OF NEXT CHAR.  (FALSE RETURN IF L1.LE.0)          SYNT2030
  705 IF(L1.LE.0) GO TO 706                                             SYNT2040
      CALL RCHXTK(J,C1)                                                 SYNT2050
      W(A1) = ICHXTK(S1,J)                                              SYNT2060
      C1 = C1+1                                                         SYNT2070
      L1 = L1-1                                                         SYNT2080
      GO TO 300                                                         SYNT2090
  706 W(A1) = 0                                                         SYNT2100
      GO TO 400                                                         SYNT2110
C  SET W(A1) TO TYPE-CODE OF PRIOR TOKEN                                SYNT2120
C  (TOKEN PRESUMABLY FOUND VIA TEST FOR TYPE IN/OUT-OF-RANGE)           SYNT2130
  710 W(A1) = ISAVE                                                     SYNT2140
      GO TO 300                                                         SYNT2150
C  SET W(A1) TO START OF REMAINDER OF STRING (AFTER THIS TOKEN)         SYNT2160
  720 W(A1) = C1                                                        SYNT2170
      GO TO 300                                                         SYNT2180
C  SET W(A1) TO LENGTH OF REMAINDER OF STRING                           SYNT2190
  730 W(A1) = L1                                                        SYNT2200
      GO TO 300                                                         SYNT2210
C  SET W(A1) TO START OF CURRENT TOKEN                                  SYNT2220
  740 W(A1) = C2                                                        SYNT2230
      GO TO 300                                                         SYNT2240
C  SET W(A1) TO LENGTH OF CURRENT TOKEN                                 SYNT2250
  750 W(A1) = L2                                                        SYNT2260
      GO TO 300                                                         SYNT2270
C  SET W(A1) TO ADDRESS OF RULE FOLLOWING CURRENT RULE                  SYNT2280
  760 W(A1) = R+NWDR                                                    SYNT2290
      GO TO 300                                                         SYNT2300
C  SET W(A1) TO F-ADDR OF CURRENT RULE                                  SYNT2310
C***  (ANY USER EXIT USED WITH THIS RULE MUST RETURN WITH U=.TRUE.)     SYNT2320
  770 CALL SYNTXF(R,Y,X(R))                                             SYNT2330
      W(A1) = F                                                         SYNT2340
      GO TO 300                                                         SYNT2350
C                                                                       SYNT2360
C  O=10 -- GO TO RULE AT (T)+W(A1)                                      SYNT2370
C  O=11 -- GO TO RULE AT *+1+W(A1)                                      SYNT2380
C  O=12 -- GO TO RULE WHOSE NUMBER IS AT (T)+W(A1)                      SYNT2390
C  O=13 -- GO TO RULE WHOSE NUMBER IS AT *+1+W(A1)                      SYNT2400
```

```
C***  NOTE.. FOR ALL OF THE ABOVE, W(A1) MUST RANGE BETW 0 AND A2        SYNT2410
C                                                                        SYNT2420
  800 CALL SYNTXF(7,Y,X(B))                                              SYNT2430
      B = T                                                              SYNT2440
      GO TO 840                                                          SYNT2450
  810 B = B+NWDR                                                         SYNT2460
  840 IF (W(A1).LT.0 .OR. W(A1).GT.A2) GO TO 490                         SYNT2470
      B = B+W(A1)*NWDR                                                   SYNT2480
      GO TO 295                                                          SYNT2490
  850 CALL SYNTXF(7,Y,X(B))                                              SYNT2500
      B = T                                                              SYNT2510
      GO TO 890                                                          SYNT2520
  860 B = B+NWDR                                                         SYNT2530
  890 IF (W(A1).LT.0 .OR. W(A1).GT.A2) GO TO 490                         SYNT2540
      B = B+W(A1)                                                        SYNT2550
      B = (X(B)-1)*NWDR+1                                                SYNT2560
      GO TO 295                                                          SYNT2570
      END                                                                SYNT2580
```

```
         REGNAM
.
.        SPECIAL FIELD UNPACKING ROUTINE DESIGNED SOLELY FOR USE BY
.        FUNCTION 'SYNTAX' TO DECODE SYNTAX RULES.
.
.        THE FIELDS ARE DEFINED AS FOLLOWS..
.
.        FLD    WORD   BIT    LEN    CODE   DESCRIPTION
.
.        1      1      00     06     O      OPERATION CODE
.        2      1      12     06     OM     OPERATION MODIFIER
.        3      1      24     06     A1     ARGUMENT OR REGISTER NO. 1
.        4      1      30     06     A2     ARGUMENT OR REGISTER NO. 2
.        5      1      18     18     A      PROCEDURE ADDRESS (O=1 ONLY)
.                                           (NOTE THAT A OVERLAPS A1 AND A2)
.        6      2      00     12     K      ACTION CODE (USED IF TRUE TEST)
.        7      2      12     12     T      TRUE ADDRESS
.        8      2      24     12     F      FALSE ADDRESS
.
.        FIELDS OF RULE 'X' ARE STORED IN LIST 'T' AS DIRECTED BY
.        CONTROL CODE 'I'.
.        IF I=1-8,
.                  SET T(I) TO CONTENTS OF FIELD 'I' IN 'X'.
.        IF I=9    SET T(1) THRU T(5)
.        IF I=10   SET T(6) AND T(7)
.
.        FORTRAN CALL IS..
.                  CALL SYNTXF(I,T,X)
.
$(0)                                   .
SYNTXF* LA      A2,*0,B11              . (A2) = I
        LA      A3,2,B11               . A3 POINTS TO X
        LA      A0,1,B11               .
        ANA,14  A0,1                   . A0 POINTS TO T-1
        SA,01   A0,ST1                 . SAVE ADDR OF T-1
        EX      RL-1,A2                . EXECUTE UNPACK INSTR (DEP ON I)
ST1     SA      A0,0,A2                . STORE 1 FIELD (FOR I=1-5 ONLY)
        J       4,B11                  .
RL      LA,12   A0,0,A3                . IF I=1-5, UNPACK JUST 1 FIELD
        LA,11   A0,0,A3                .
```

```
         LA,9     A5,0,A3         .                                   SYNT0410
         LA,8     A5,0,A3         .                                   SYNT0420
         LA,7     A5,0,A3         .                                   SYNT0430
         J        T6              . GET FIELD 6 UNSIGNED              SYNT0440
         J        T7              . GET FIELD 7 UNSIGNED              SYNT0450
         J        T8              . GET FIELD 8 UNSIGNED              SYNT0460
         J        T15             . GET FIELDS 1-5 AT ONCE            SYNT0470
         J        T67             . GET FIELDS 6 AND 7 AT ONCE        SYNT0480
T6       LMJ      A1,F6           .                                   SYNT0490
         J        4,B11           .                                   SYNT0500
T7       LMJ      A1,F7           .                                   SYNT0510
         J        4,B11           .                                   SYNT0520
T8       LA       A5,1,A3         . GET FIELD 8                       SYNT0530
         LSSL     A5,24           .                                   SYNT0540
         SZ       A4              .                                   SYNT0550
         LDSL     A4,12           .                                   SYNT0560
         LSSL     A4,1            .                                   SYNT0570
         ANA,14   A4,1            . RETURN FIELD 8 TIMES 2 MINUS 1    SYNT0580
         SA       A4,8,A0         .                                   SYNT0590
         J        4,B11           .                                   SYNT0600
F6       LA       A5,1,A3         . GET FIELD 6 AS IS                 SYNT0610
         SZ       A4              .                                   SYNT0620
         LDSL     A4,12           .                                   SYNT0630
         SA       A4,6,A0         .                                   SYNT0640
         J        0,A1            .                                   SYNT0650
F7       LA       A5,1,A3         . GET FIELD 7 TIMES 2 MINUS 1       SYNT0660
         LSSL     A5,12           .                                   SYNT0670
         SZ       A4              .                                   SYNT0680
         LDSL     A4,12           .                                   SYNT0690
         LSSL     A4,1            .                                   SYNT0700
         ANA,14   A4,1            .                                   SYNT0710
         SA       A4,7,A0         .                                   SYNT0720
         J        0,A1            .                                   SYNT0730
T15      LA,12    A4,0,A3         . GET FIELDS 1-5                    SYNT0740
         SA       A4,1,A0         .                                   SYNT0750
         LA,11    A4,0,A3         .                                   SYNT0760
         SA       A4,2,A0         .                                   SYNT0770
         LA,9     A4,0,A3         .                                   SYNT0780
         SA       A4,3,A0         .                                   SYNT0790
         LA,8     A4,0,A3         .                                   SYNT0800
```

```
        SA     44,4,[illegible]        .                SYNT0810
        LA,1   44,[illegible]          .                SYNT0820
        SA     44,[illegible]          .                SYNT0830
        J      4,B1[illegible]         .                SYNT0840
T67     L[illegible]   41,F6                            SYNT0850
        L[illegible]   4[illegible],F7 .                SYNT0860
        J      4,B1[illegible]         .                SYNT0870
        END                                             SYNT0880
```